# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | 1 September 1994 | Final Report 1/6/93 - 1/6/96 |

**4. TITLE AND SUBTITLE**

The Detection and Extraction of Features of Low Probability of Intercept Signals Using Quadrature Mirror Filter Bank Trees

**5. FUNDING NUMBERS**

F49620-93-1-0404
3484/2S 61103D

**6. AUTHOR(S)**

Glenn E. Prescott
Thomas C. Farrell

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Telecommunications & Information Sciences Laboratory
Department of Electrical Engineering & Computer Science
University of Kansas, Lawrence, KS 66045

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Air Force Office of Scientific Research
110 Duncan Avenue, Suite B115
Bolling AFB, DC 20332

AFOSR-TR-96

nm

0459

**11. SUPPLEMENTARY NOTES**

Distribution Unlimited

19961016 082

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

UL

DTIC QUALITY INSPECTED 4

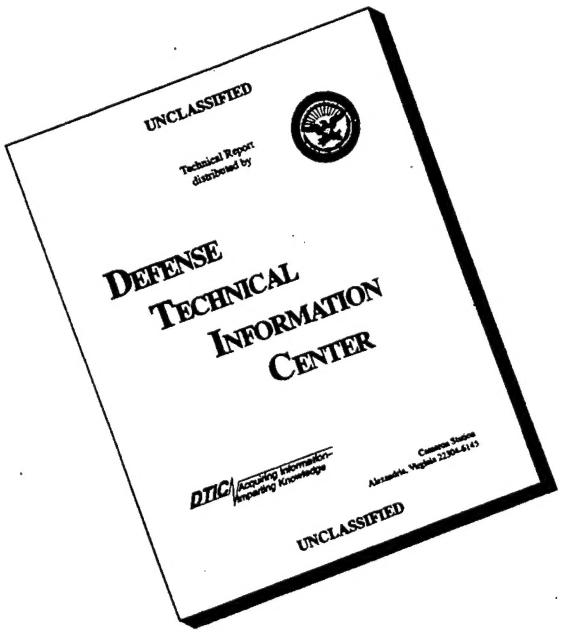
**13. ABSTRACT** *(Maximum 200 words)*

A new type of spread spectrum intercept receiver is described which uses orthogonal Wavelet techniques and a Quadrature Mirror Filter (QMF) bank tree to decompose a waveform into components representing the energy in rectangular "tiles" in the time frequency plane. By simultaneously examining multiple layers of the tree, the dimensions of concentrations of energy can be estimated with a higher resolution than is normally associated with linear transform techniques. This allows detection and feature extraction even when the interceptor has little knowledge of specific parameters of the signal being detected. In addition, the receiver can intercept and distinguish between multiple signals. For each category of spread spectrum, the receiver estimates the energy cells' positions in the time frequency plane, the cells' bandwidths, time widths and signal to noise ratios, and the energy distribution within each cell. With this information, a classifier can then determine how many transmitters there are, and which cells belong to each. In this report, algorithms are described for detecting and extracting features for each of the spread spectrum signal formats. These algorithms are analyzed mathematically and the results are verified with simulation. The detection abilities of these algorithms are compared with other spread spectrum detectors hat have been described in the literature.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Multiresolution, LPI, Signal Detection, Wavelets, Quadrature Mirror Filters | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | |

# DISCLAIMER NOTICE




**THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.**

# Final Technical Report

to

Air Force Office of Scientific Research
AFOSR/NM (Mathematics & Signal Processing)
Bolling AFB, Washington DC 20332-6448

AFOSR Grant #F49620-93-1-0404

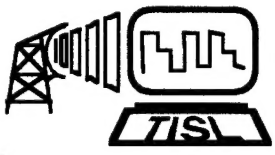
# Detection and Extraction of Features of Low Probability of Intercept Signal Features Using Quadrature Mirror Filter Banks

Period of Performance 1 June 1993 - 31 May 1996

Thomas C. Farrell

Glenn E. Prescott
Principal Investigator
Associate Professor of Electrical Engineering

June 1996




Telecommunications and Information Sciences Laboratory
The University of Kansas Center for Research, Inc.
2291 Irving Hill Drive, Lawrence, Kansas 66045

## Table of Contents

## List of Figures

## List of Tables

# The Detection and Extraction of Features of Low Probability of Intercept Signals Using Quadrature Mirror Filter Bank Trees

Thomas C. Farrell, B.S./M.S.
Department of Electrical Engineering and Computer Science, 1996
University of Kansas

**Abstract**

This research describes a new type of spread spectrum intercept receiver. The receiver uses orthogonal Wavelet techniques and a Quadrature Mirror Filter (QMF) bank tree to decompose a waveform into components representing the energy in rectangular "tiles" in the time frequency plane. By simultaneously examining multiple layers of the tree, the dimensions of concentrations of energy can be estimated with a higher resolution than is normally associated with linear transform techniques.

This allows detection and feature extraction even when the interceptor has little knowledge of specific parameters of the signal being detected. For example, no prior knowledge of channelization or time segmentation is assumed. In addition, the receiver can intercept and distinguish between multiple signals. The spread spectrum formats considered are: Fast Frequency Hopping (FH), Time Hopping (TH), Fast FH/TH, Direct Sequence (DS), Fast FH/DS, TH/DS, Fast FH/TH/DS, and Slow FH. For each of these, the receiver estimates the energy cells' positions in the time frequency plane, the cells' bandwidths, time widths, and signal to noise ratios, and the energy distribution within each cell. With this information, a classifier can then determine how many transmitters there are, and which cells belong to each.

In this research, algorithms are described for detecting and extracting features for each of the signal formats. These algorithms are analyzed mathematically and the results are verified with simulation. The detection abilities of these algorithms are compared with other spread spectrum detectors that have been described in the literature.

# The Detection and Extraction of Features of Low Probability of Intercept Signals Using Quadrature Mirror Filter Bank Trees

## I. Introduction

Spread Spectrum signals are often used in the military environment to provide Low Probability of Intercept (LPI), or covert, communications. Recently, spread spectrum techniques have also been incorporated into such civilian applications as wireless local area networks and cellular telephones, where the primary advantages are low transmitter power requirements and low probability of interference. As the use of these techniques becomes more widespread, so do the requirements for people, other than the intended receiver, to detect and determine key features of the signals. Two examples of this are the requirement to police the electromagnetic spectrum, and the need for field engineers to determine how much traffic a particular Spread Spectrum band carries in a particular environment.

Most descriptions of Spread Spectrum detection receivers published in the open literature begin with an assumption that the interceptor knows just about everything there is to know about the signal to be intercepted except for its presence, and the pseudo-random sequence used to spread (and/or hop) the signal [15] [45]. In particular, the interceptor is generally assumed to know the channelization, time duration, and hop synchronization of frequency hopped (FH) and time hopped (TH) signals, and the bandwidth of direct sequence (DS) signals. In addition, the assumption is often made (sometimes implicitly) that, at most, one signal will be present.

In many military and civilian environments, these assumptions are not appropriate. For example, an FH LPI military radar might not be turned on until necessary, depriving the enemy of channelization information until it's too late. It is also unrealistic to assume only one LPI signal will be present in most military situations, where both sides will have radar and communications systems operating. In the civilian world, it is easy to imagine a large office building, for example, with dozens (perhaps hundreds) of wireless local area networks. There is, therefore, a need for an intercept receiver that does not depend on a detailed knowledge of signal parameters, and that can distinguish between, and categorize, signals operating in a busy environment.

This research presents a new type of LPI intercept receiver--one based on a linear decomposition of the received waveform via a Quadrature Mirror Filter (QMF) bank tree with Wavelet filters. This receiver provides good detection performance even when much of the structure of the LPI signal is unknown. In addition, this receiver is able to extract certain features of the LPI signal and distinguish between multiple signals.

As used in this report, "Detection" refers to the process, by the interceptor, of determining whether spread spectrum signals are present. It is a binary decision: either the interceptor decides no signals are present, or decides one or more signals are present. In the detection process, information is not necessarily obtained about how many signals are present, or the characteristics of the signal(s). "Feature extraction" on the other hand, as used in this report, will refer to estimating bandwidths, hop rates, hop synchronization, and signal to noise ratios (SNRs)--features that may be used to characterize and distinguish between signals.

Also, in this report, the term "waveform" will generally refer to whatever the interceptor is assumed to receive, which will usually be white Gaussian noise (WGN) that may, or may not, include

a man made LPI spread spectrum signal. The term "signal", on the other hand, will usually refer specifically to the spread spectrum signal. (Any exceptions will either be clear from the context, or explicitly noted.)

This chapter begins with a brief background of the LPI signal interception problem, including a brief description of the LPI signals and intercept receivers that have been presented in the open literature. In this section, time frequency decomposition methods are also discussed. The problem statement is then laid out and the approach and scope of the research is described. Assumptions used in the research are listed, and, finally, a brief overview is given, describing the organization of the remainder of the report.

## Background

### LPI Signals

Knowledge of the specific formats for the LPI signals are important for determining detection algorithms, and algorithms for extracting some of the key features. Although there are many possibilities, certain LPI signal structures are most often described in the literature [16] [17] [38]. The ones considered in this research are:

1) **Fast Frequency Hopping (FH)**. For this signal, the transmitter rapidly hops a carrier (pseudo-randomly) among a large number of center frequencies. The bandwidth of each hopped portion of the signal is determined by the hop rate. In this report, each hop will be referred to as a "cell". The energy distribution of each cell has a sinc-squared shape in the frequency dimension (assuming fairly sharp hops with constant energy output in the time dimension). If the hop duration is T, it is common to take the cell bandwidth to range from the hop center frequency minus 1/2T to the hop center frequency plus 1/2T. This gives each cell a time bandwidth product of unity, and includes most of the cell's energy.

2) **Time Hopping (TH)**. Similar to fast FH, except the carrier frequency is constant and the transmitter only transmits during one of several time slots (selected pseudo-randomly).

3) **Fast FH/TH**. A combination of the two techniques described above.

In all three of these signal structures the cells' time-bandwidth products will be unity. These are discussed in more detail in Chapter V.

4) **Direct Sequence (DS)**. For this signal, the transmitter modulates the information with a higher frequency pseudo-random waveform (known to the intended receiver). The effect is to spread the signal over a much wider bandwidth than it would otherwise occupy. The pseudo-random waveform is often a random binary waveform, spreading the signal energy under a sinc-squared envelope. This signal structure is discussed in more detail in Chapter VI.

5) **Fast FH/DS**. For this signal, each fast FH cell is further spread (in frequency) with a pseudo-random waveform. This creates cells with time-bandwidth products much greater than unity.

6) **TH/DS** and **Fast FH/TH/DS**. A combination of techniques described above. The cells' time-bandwidth products will be much greater than unity.

7) **Slow FH**. In this case, the carrier signal is first modulated by the information using multiple frequency shift keying (MFSK). The transmitter then hops the signal relatively slowly, so each cell's bandwidth is determined by the MFSK waveform. The cell's time-bandwidth product will be greater than unity.

Signal structures 5) through 7) are discussed in more detail in Chapter VII.

The transmitter's goal in each of these cases is to "hide" the signal by increasing the bandwidth over which an interceptor must search. The intended receiver is assumed to have the pseudo-random waveform, and thus has a much smaller bandwidth to consider. This allows the transmitter to use relatively low energy levels, so the interceptor must expect to see a fairly low SNR.

**Summary of Current Knowledge**

**LPI Energy Detection.** As with the receiver proposed in this report, most of the LPI intercept receivers discussed in the open literature are based on some form of energy detection.

The receiver most often discussed for the detection of DS signals is the radiometer (energy detector) [16] [41] [42]. This receiver has an input filter matched to the DS signal. The filter's output is then squared and integrated for the duration of the observation. If X is a random variable specifying the radiometer's output and $N_0$ is the single sided noise spectral density, we can define $X' = 2X / N_0$. When WGN alone is present in the radiometer's input, the probability distribution function (pdf) for X' has a Chi Squared distribution. When energy from a deterministic signal and WGN are present, the distribution is Chi Squared with a non-centrality parameter [41]. By comparing the radiometer's output against a threshold, a detection decision can be made and, because of the nature of the pdf curves, the probability of detection will always exceed the probability of false alarm. The radiometer detector is described in more detail in Chapter V, where it is shown to be a good receiver of last resort. It does the best job of detecting a signal embedded in WGN when no assumptions about the signal structure can be made, except for the maximum bandwidth and the maximum time interval during which the signal may be turned on.

For FH signals, detectors described in the literature consist of banks of radiometers, with each input filter matched to a hop channel, and the integration interval matched to the cell duration [16] [41] [45]. (Obviously, the interceptor must know the channelization, cell duration, and hop synchronization to use these types of detectors.) There are several possible ways to combine the radiometer outputs. The optimal method, for Fast FH, TH, and Fast FH/TH signals (and for Fast FH/DS and Fast FH/TH/DS signals when the energy distribution due to direct spreading is assumed to be unknowable), is derived in Appendix A of [16]. There, a maximum likelihood test statistic is found which can be compared against a threshold to make a detection decision. Levitt, et al, in [30], claim this architecture is not optimal for Slow FH signals because energy is not uniformly distributed within the cells, and derive the optimal detector results for that signal.

A disadvantage of this detector is that the test statistic is based on the potential signal's cell energy at the interceptor. A simpler, somewhat sub-optimal, technique is to compare the output of each radiometer to a threshold, and to declare a signal present for a time slot if any outputs exceed the threshold. This type of receiver is usually referred to as a Filter Bank Combiner (FBC) and is described in [16] [18] [45]. [18] also discusses the effects of a mismatch between the hop times and the interceptor's time slots on this type of receiver.

Several recent papers have described improvements on the basic FBC. [35] assumes the interceptor has knowledge of the statistical frequency of a hop occurring in any particular channel, and, based on this, adapts each channel's threshold to reduce the number of false alarms. [19] suggests using an Artificial Neural Network (ANN) in place of each channel's threshold detector and making a detection decision based on all of the radiometers' outputs.

Table I summarizes the knowledge an interceptor may have about an LPI signal, and the types of intercept receivers described in the literature that exploit that knowledge. As the table shows, there is a blank space for the particular case when the interceptor knows the signal structure, and therefore knows how the signal cells' energy (or DS signal energy) is concentrated in both time and frequency, but does not know the specific bandwidth, channelization, hop rate, or hop synchronization. The receiver described in this report improves on the radiometer for this situation, by taking advantage of the concentrations of signal energy. This new type of receiver uses some recent advances in the state of knowledge in time frequency decomposition, and so, before presenting the receiver architecture itself, it is necessary to briefly review the literature.

**Time Frequency Decomposition.** Much has been published, particularly in recent years, on various methods of decomposing a waveform and displaying it as a function on the time frequency plane. Broadly, the most common of these methods can be divided into linear and bilinear transforms, with the Short Time Fourier and Wavelet Transforms being examples of the former, and the Wigner Transform being an example of the latter.

A general bilinear transform, sometimes referred to as the "Cohen Distribution" after the author who showed it is a super set of many bilinear transforms that were developed independently, is [11]

$$a(t,f) = \frac{1}{4\pi^2}\iiint e^{-j\theta t - j\tau 2\pi f + j\theta u}\,\Phi(\theta,\tau)\,x^*\left(u-\frac{1}{2}\tau\right)x\left(u+\frac{1}{2}\tau\right)du\,d\tau\,d\theta \tag{1.1}$$

where

| | |
|---|---|
| t and f | refer to the time and frequency coordinates in the plane |
| $x$ | is the input signal |
| $x^*$ | is the complex conjugate of the input signal (and would be identical to x for the real signals we consider in this research) |
| $\Phi(\theta,\tau)$ | is an arbitrary function called the kernel. For the Wigner distribution, it is unity. |

These transforms are called bilinear because the input waveform appears twice. This allows better resolution in the time frequency plane than the linear techniques, but greatly increases the computational burden and results in other side effects. For example, with the Wigner distribution, a signal that consists of a certain tone for a finite time, zero for a time, and then another tone of a different frequency, will display positive values for $a(t,f)$ where expected for the two tones, but will also display positive values for a frequency in between the two true frequencies for the time interval between the tones [11]. These "cross-terms" are due to the non-linear nature of the transform.

Many papers have been published showing how bilinear transforms may be used in signal detection. For example, [20] addresses the detection of non-stationary Gaussian signals in additive WGN when the expected value of the Wigner distribution of the signal is known, and determines a test statistic that can be compared to a threshold. [25] develops a similar approach to detect linear

**Table I**

Type of Intercept Receiver to Use, Based on the
Interceptor's Knowledge of the LPI Spread Spectrum Signal

| **Elements of Spread Spectrum Signal Known to Interceptor** | **Intercept Receiver to Use** |
|---|---|
| Region of Time Frequency Plane Containing Signal | Radiometer/Threshold Detector |
| Information Above and:<br>Structure (DS, FH, TH, FH/TH, etc.) | |
| All Information Above and:<br>Channelization[1]<br>Cell Bandwidth[1]<br>Hop Rate/Cell Duration[1]<br>Hop Synchronization[1] | Filter Bank Combiner |
| All Information Above and:<br>Received Cell Energy[1] | "Optimal Detector" |
| All Information Above and:<br>Pseudo-Random Spreading/Hop Code[2] | Spread Spectrum Receiver |

Notes:

[1] Hopped signal structures only
[2] Generally known only by the intended receiver(s)

chirp FM signals. In there, the Wigner distribution of the received waveform is integrated along all possible lines in the time frequency plane, and the largest result is assumed to contain the chirp. [6] discusses the cross-Wigner distribution, where the conjugate of the received signal in (1.1) is replaced by the desired signal, and uses it to distinguish the acoustic signatures of diesel engine cylinders.

However, very little has yet been published specifically using these transforms for the detection of LPI waveforms. This is probably because bilinear transforms have only recently appeared in the communications literature, and there are a relatively small number of people working in LPI research.

Due to the computational complexity and the possibility of confusing cross-terms, the bilinear transform will not be considered further in this research. Rather, linear transforms will be found to have all of the properties required. Linear transforms have the following form

$$a_k = \int f(t)\Psi_k(t)dt \tag{1.2}$$

where:

| | |
|---|---|
| $\Psi_k(t)$ | is the basis set |
| t | is the time index |
| k | is the function index |

The Fourier Transform, for example, has a basis set consisting of sines and cosines of frequency $2\pi k$. The transforms are said to be orthonormal if we restrict the basis functions so [28]

$$\int \Psi_k^*(t)\Psi_m(t)dt = \begin{cases} 1 & k = m \\ 0 & k \neq m \end{cases} \tag{1.3}$$

It is possible to sample the input waveform at the Nyquist rate and retain all of the information [32] [36]. In that case, the time variable, t, in (1.2) and (1.3) should be taken to be discrete, and the integrals should be replaced with summations. This is the case discussed in Chapter II. Here, it is shown that when the waveform consists entirely of WGN, the squares of the coefficients will have random values with a Chi Squared probability distribution. When a deterministic signal is added, certain coefficients (depending on the signal energy distribution) will have probability distributions that are Chi Squared with non-centrality parameters and will, therefore, tend to have larger mean values, making threshold detection a possibility.

Recently, Wavelet bases have gained prominence in the communications literature [10] [12] [24] [44]. These bases are effectively non-zero for only a finite time interval, and can be designed to satisfy (1.3). In [24] [44] it is shown these orthogonal Wavelets can be implemented using QMFs: Filter pairs designed to divide the input signal energy into two orthogonal components based on the frequency.

The Wavelet Transform divides the time frequency plane as shown in Figure 1.1, where each squared coefficient represents (approximately) the energy within "tiles" (rectangular regions) of the time frequency plane. A characteristic of Wavelet transforms is that the tiles become shorter in time and occupy a larger frequency band as the frequency is increased. However, using Wavelet techniques to develop an appropriate basis set, and a QMF bank for implementation, it is shown in [24] [44] that it is possible to decompose the waveform in such a way that tiles have the same dimensions regardless of frequency. Because the transform is linear, there is a fundamental limit on the minimum area of each of these tiles. However, the nature of the QMF bank configuration is such that each layer outputs a matrix of coefficients for tiles that are twice as long (in time) and half as tall (in frequency) as the tiles in the previous layer. By properly comparing these matrices, it is possible to deduce signal features with both fine frequency and fine time resolutions.

Using these techniques, then, it is possible to decompose a waveform and estimate the bandwidths, center frequencies, energy distribution, and SNRs of DS signals, and the bandwidths, durations, locations in the time frequency plane, and SNRs, of individual FH and TH cells. In addition, for signals using a combination of techniques such as FH/DS or TH/DS, it is possible to determine the energy distribution within individual cells. All of this information, of course, can then

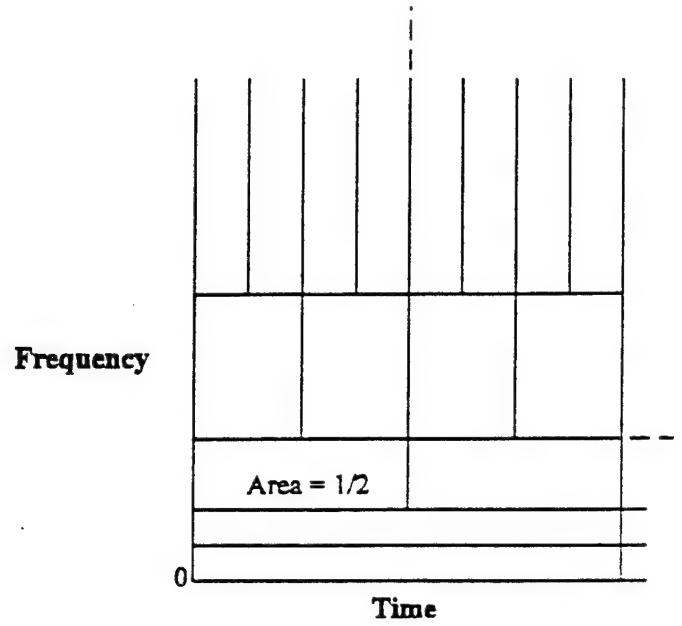



Figure 1.1. Time Frequency Diagram For the Wavelet Transform

be used by the interceptor to decide how many transmitters, and which types, are in operation. Evidently, nothing has yet been published specifically using Wavelet, or related, transforms for this sort of analysis. In particular, no papers have yet been published in which information is simultaneously derived from multiple layers of the QMF bank, as described in the last paragraph. This, again, is presumably due to the relative newness of the material and the number of researchers in LPI detection.

Some material, however, has been published using linear transforms to detect more general classes of signals. [4], for example, describes (in words) the use of arbitrary orthogonal basis functions in the detection of signals. [37], develops a fairly general mathematical framework to detect transient signals (where "transient" is used to mean a signal that begins and ends within the observation time).

Several papers have been published in recent years on the design of QMFs [1] [7] [8] [9] [26] [27] [31]. The primary uses of QMFs, described in these papers and in [12] [24], are for sub-band coding and non-stationary signal compression. It can be shown these filters have inverses which lead to theoretically perfect reconstruction of the input waveform. To compress a sequence that has most of its energy in certain frequency bands, then, one can filter it with a digital QMF bank, and only transmit output sequences with significant energy components.

**Cyclostationary Detection.** This section would not be complete without mentioning another type of LPI intercept receiver published in the literature; one that uses a technique different from energy detection [22]. For this receiver, the LPI signal is modeled as a cyclostationary random process. The spectral correlation of the input waveform is computed and examined for periodicities in the signal (hop rate, for example). It is claimed this technique is superior to energy detection,

particularly in conditions of varying background noise, and when more than one LPI signal is present [22] [23].

### Problem Statement

This research will develop methods to decompose a waveform, using orthogonal basis functions and a QMF bank tree, and extract detailed information about embedded LPI signals.

### Approach and Scope

The proposed receiver's block diagram is shown in Figure 1.2. A received waveform is band-pass filtered and sampled at the Nyquist rate. The digital sequence is then fed to the QMF bank tree where it is decomposed, and matrices of weights are output from each layer. These weights are then squared to produce coefficients representing the energy in each portion of the waveform.

This information is then analyzed to determine: 1) Where FH and TH cells (if any) are located, their dimensions, SNRs, and internal energy distributions, 2) Where DS signals are located, their bandwidth, SNRs, and internal energy distributions. The output of the analyzer is a list of cells and DS signals, and their parameters.

The classifier then takes the list from the analyzer, determines which cells apparently belong to common transmitters, and outputs a list of transmitters, their types, and parameters. In a real world receiver, further "filtering" could be done at this point, eliminating probable false alarms and signals that are not of interest to the interceptor. The classifier may even be adaptive--changing classification criteria based on current and previous input and results.

The research reported on in this report is primarily concerned with the receiver's QMF bank tree and the analyzer block. The ideal QMF would be maximally flat in the regions of interest, and would absolutely reject all signal energy elsewhere. Since this is mathematically impossible, some research went into finding the "best" filter shape for this application. For the analyzer block, the research focused on developing algorithms to detect and determine the features for each of the types of LPI signals.

This research did not look at the classifier function in any great detail, although it trespassed in certain limited areas. In particular, it was necessary to set criteria for rejecting false alarms in order to make valid comparisons with other detectors. The primary job of the classifier, however, is to take the list of cells and parameters, determine the most likely number of transmitters, and group the cells to the transmitters. This specific task was not a part of this research.

### Assumptions

Background noise is always assumed to be additive WGN (bandlimited, when considered past the initial bandpass filter).

The only signals assumed to be present are the LPI signals described above.

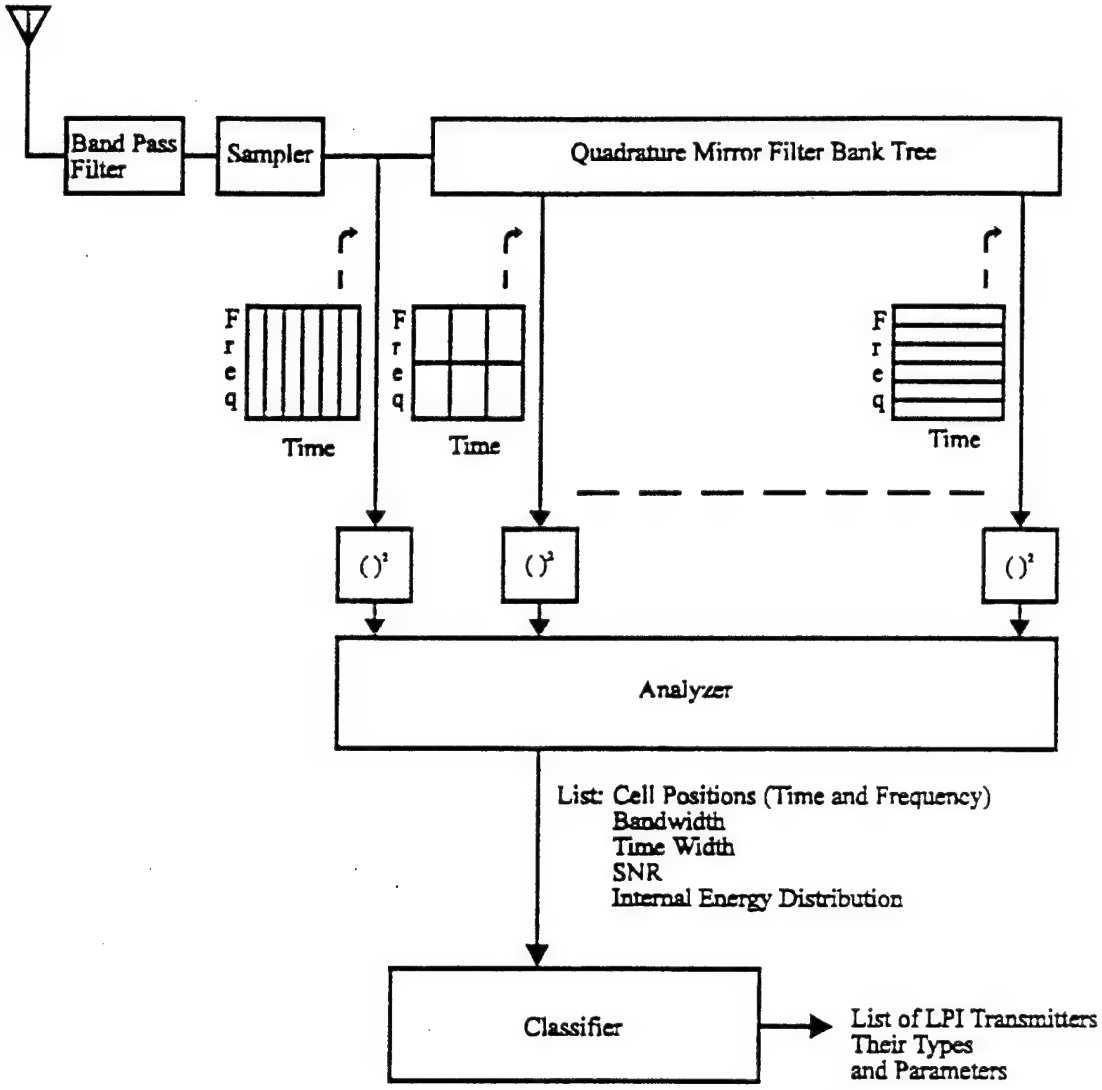



Figure 1.2. LPI Receiver Block Diagram

The input bandpass filter and sampler in the receiver shown in Figure 1.2 are assumed to be ideal.

Fading, and other adverse effects (except for the addition of WGN) in the communications channel are assumed to be negligible.

Simple, non-adaptive, thresholds are used in lieu of the classifier to make detection decisions.

## Overview of the Report

The rest of this report is primarily concerned with two parts of the receiver architecture: The QMF bank tree, and the analyzer block. Chapters II and III deal with the former, while Chapters V through VIII discuss algorithms used in the latter.

Chapter II discusses the mathematical background for signal energy detection in WGN using Wavelets and other orthogonal signal decompositions. Here, the basic structure for the QMF bank tree is laid out. Although most of the material in this chapter is an interpretation of material already published in the literature, it is background necessary for what follows.

Chapter III uses the mathematics developed in Chapter II and the particular requirements of the intercept receiver to find several candidate sets of filters for the QMF bank tree.

Mathematical analysis is used wherever possible in this research. However, much of the performance analysis turns out to be intractable. Therefore, Monte-Carlo simulation is resorted to at certain points. The (Matlab) code used to generate the spread spectrum signals and to decompose the waveforms in the manner of the QMF bank tree is described in Chapter IV and listed in Appendix A. The code used to simulate the detection and feature extraction algorithms presented in Chapters V through VIII are listed and described in Appendices B through E. These are included for the sake of the reader interested in conducting further research, as well as to help clarify details of the methods used.

Chapter V begins by describing in detail the structure of the cells of FH, TH and FH/TH signals. The chapter then presents more background, describing in detail the radiometer/threshold detector, as well as the optimal receiver and FBC. This is used as a foundation to develop a new detection algorithm for signals whose cells have time bandwidth products of one. From this algorithm, in turn, a feature extraction algorithm is developed to determine the cells' bandwidths, duration, energy, and position in the time frequency plane.

Chapter VI describes the DS signal in detail, and then develops a detection algorithm for this type of signal. The algorithm is also used to estimate signal features important to the interceptor: The signal energy, center frequency, and bandwidth.

Chapter VII deals with hopped signals with time bandwidth products greater than one: FH/DS, TH/DS, FH/TH/DS, and Slow FH signals. The structure of these signals are described in detail, and a detection algorithm is developed. Then an algorithm is developed to estimate the cells' bandwidths, duration, energy, and position in the time frequency plane.

Finally, Chapter VIII looks at using a least squares algorithm to distinguish cells based on the distribution of their energy.

## II. The Detection of Signal Energy In Noise Using Wavelets and Related Techniques

### Introduction

In this chapter we consider an issue of detection. Specifically: given a waveform, most of whose energy comes from stationary white Gaussian noise (WGN), is a deterministic signal present? We explore a way to answer the question by looking for concentrations of the waveform energy distributed in a non-random manner.

We do this by decomposing the waveform into a sum of terms, with each term consisting of a function from a basis set and a coefficient multiplier. A common example of this is the Fourier Transform, where a function is decomposed into a sum of weighted sinusoids. Another example, as we will see, is the Wavelet Transform.

The goal is to find a basis set that will concentrate the deterministic signal energy in as few terms as possible. A detection decision may then be made by comparing the values of the coefficients against a threshold value. If any are above the threshold, a signal is considered to be present; if not, absent. Of course, since the noise is random, there is no way to make a perfect decision. For a given signal energy to noise energy ratio (SNR), a given detection scheme and a certain threshold will have a given probability of detection and probability of false alarm.

Here, we will work strictly with discrete time waveforms (except for a brief digression to compare the energy in the continuous waveform to the energy in it's sampled counterpart). We lose nothing by doing this, since it can be shown all of the information in the waveform can be represented by properly sampling the signal [28]. This process includes filtering before sampling to remove much of the white noise (but, hopefully, none of the signal). Since this is the case, after sampling we will be dealing with "band limited white Gaussian noise," a term which refers to Gaussian noise that is band limited but has a (nearly) flat spectral density curve over the pass band [46]. Sampling is important in practice, since much of the processing discussed in this paper is best done digitally. We should also note the basis functions themselves may either be defined only for discrete times, or defined for all time, but only evaluated at discrete times.

In the next section, we will develop a framework for the general decomposition of waveforms, and discuss some properties of interest. This development is similar in some ways to that used in many engineering texts specifically for the Fourier Transform (see, for example [28] [32] [36] [46]). Here, however, we consider a general basis set and, subject to a few conditions, find properties common to all. We then look at some examples of commonly used basis sets and their characteristics. Then, we look specifically at the Wavelet transform, its development, characteristics, implementation, and advantages in detecting certain types of signals. Finally, we look at a generalization of Wavelet Transform techniques that offer further possibilities for detection.

### General Framework For the Decomposition of Waveforms

#### The Basis Set

Given a waveform, f(n), we wish to decompose it as follows

$$f(n) = \sum_k a_k \Psi^*_{kn} \tag{2.1}$$

where:

| | |
|---|---|
| $\Psi_{kn}$ | is the basis set (* denotes complex conjugation) |
| n | is the time index $n \in$ subset of integers |
| k | function index $k \in$ subset of integers |

By examination of (2.1), the first requirement for a good basis set becomes obvious: it must span the space of possible waveforms. In other words, the waveform (actually, the deterministic signal in the waveform) must be able to be written, at least to some suitable approximation, as a weighted sum of the basis functions. With the Fourier basis set of sinusoids, for example, it can be shown the mean square error of any signal we encounter in practice will approach zero as the number of basis functions approaches infinity [28].

We next place the restriction of orthonormality on the basis functions

$$\sum_n \Psi^*_{kn} \Psi_{mn} = \begin{cases} 1 & k = m \\ 0 & k \neq m \end{cases} \tag{2.2}$$

By taking (2.1) and adding the summation, over time, of one of the basis functions, we can find a general formula for the coefficients

$$\begin{aligned} \sum_n f(n)\Psi_{mn} &= \sum_n \sum_k a_k \Psi^*_{kn} \Psi_{mn} = \\ &\sum_k a_k \sum_n \Psi^*_{kn} \Psi_{mn} = a_m \end{aligned} \tag{2.3}$$

where the last step follows from orthonormality. We are now in a position to see why this decomposition is important for us. The energy of the waveform is the sum, over time, of the values squared [36] [32]

$$\begin{aligned} \sum_n |f(n)|^2 &= \sum_n \Big(\sum_k a_k \Psi^*_{kn}\Big)\Big(\sum_m a^*_m \Psi_{mn}\Big) = \\ &\sum_n \Big(\sum_k \sum_m a_k a^*_m \Psi^*_{kn} \Psi_{mn}\Big) = \\ &\sum_k \sum_m \Big(a_k a^*_m \sum_n \Psi^*_{kn} \Psi_{mn}\Big) = \\ &\sum_k |a_k|^2 \end{aligned} \tag{2.4}$$

where, once again, the last step is due to orthonormality. This result, sometimes called Parceval's theorem, is critical for detection. Since we are looking for signal energy, we can apply the basis set to the waveform, as in (2.3), and then consider the values of the coefficients squared. A good basis set, for our purposes, will place the signal energy in as few of the coefficients as possible. A detector who knows what he is looking for can then disregard the other coefficients, reducing the probability of

false alarm. If the Fourier basis set is used, for example, a detector can filter out frequency components that could not contain the signal.

Incidentally, any basis set meeting (2.4) is sometimes called a "tight frame" in the mathematical literature. If we started off by declaring that to be our requirement for a basis set, formula (2.1), reconstructing f(n), would necessarily follow [10]. For our purposes, the basis set must form an invertible transform.

**Energy of the Discrete Waveform**

In (2.4) we call the sum of the squares of the waveform values the energy of the discrete signal. If the signal given to us is discrete in nature, we can accept this as a definition. If, however, we are sampling a continuous time signal, we must relate (2.4) to the original signal's energy. To do this, let's assume we have a band limited continuous signal, $f_a(t)$, that we sample at the Nyquist rate, or faster

$$f(n) = f_a(nT) \tag{2.5}$$

where T is the time between samples. To reverse (2.5) we can use the interpolation formula [36] [32]

$$f_a(t) = \sum_n f(n)\mathrm{sinc}\left[(t-nT)/T\right] \tag{2.6}$$

Where the sinc function is

$$\mathrm{sinc}(k) = \begin{cases} \dfrac{\sin(\pi k)}{\pi k} & k \neq 0 \\ 1 & k = 0 \end{cases} \tag{2.7}$$

Now, the energy of the continuous time function is [28]

$$\varepsilon = \int_{-\infty}^{\infty} |f_a(t)|^2 dt \tag{2.8}$$

Using (2.6), we find

$$|f_a(t)|^2 = \left[\sum_n f(n)\mathrm{sinc}\left[(t-nT)/T\right]\right]\left[\sum_p f(p)^*\mathrm{sinc}\left[(t-pT)/T\right]\right] =$$

$$\left[\sum_n |f(n)|^2\left[\mathrm{sinc}\left[(t-nT)/T\right]\right]^2\right] + \tag{2.9}$$

$$\sum_n \sum_{p\neq n} f(n)f(p)^*\left[\mathrm{sinc}\left[(t-nT)/T\right]\right]\left[\mathrm{sinc}\left[(t-pT)/T\right]\right]$$

and integrating gives us

$$\varepsilon = \int_{-\infty}^{\infty} |f_a(t)|^2 dt = \sum_n |f(n)|^2 \int_{-\infty}^{\infty} \left[\text{sinc}\left[(t-nT)/T\right]\right]^2 dt +$$

$$\sum_n \sum_{p \neq n} f(n) f(p)^* \int_{-\infty}^{\infty} \left[\text{sinc}\left[(t-nT)/T\right]\right]\left[\text{sinc}\left[(t-pT)/T\right]\right] dt = \tag{2.10}$$

$$T \sum_n |f(n)|^2$$

where the result comes from the fact that the integral of the sinc squared function is T, and the integral of the two sinc functions are zero, when n and p are integers that are not equal. (2.10) tells us our definition of the discrete signal's energy is proportional to the continuous signal's energy. Often, T is normalized to one, making the two energies equal.

Notice (2.10) does not necessarily imply $|f(n)|^2$ for a particular n represents the energy of the analog waveform around $f_a(nT)$. Ideally we would like

$$\int_{n_0T-\frac{T}{2}}^{n_0T+\frac{T}{2}} |f_a(t)|^2 dt \approx T|f(n_0)|^2 \tag{2.11}$$

to be an exact equality, but of course, as Figure 2.1 shows, it will only be approximately true in general. In fact, the error will be on the order of [39]

$$O\left( T^2 \frac{d}{dt} |f_a(t)|^2 \Big|_{t=n_0T} \right)$$

which we see can be minimized by decreasing T. This can be accomplished either by actually sampling the analog waveform more often, or by retaining the original sampling rate and using (2.7) to interpolate and increase the size of the sequence.

**Statistical Properties of the Coefficients**

Since we are going to measure the values of the coefficients of (2.3), we need to know their statistical properties. First, let's consider a waveform consisting entirely of band limited WGN with a mean of zero and a variance of $\sigma^2$, which we will denote by $f(n) = \eta(n) \sim N(0,\sigma^2)$. Also, we will assume the value for each n is statistically independent of other values. The mean, or expected value of each of the coefficients is

$$E[a_k] = \sum_n E[\eta(n)]\Psi_{kn} = 0 \tag{2.12}$$

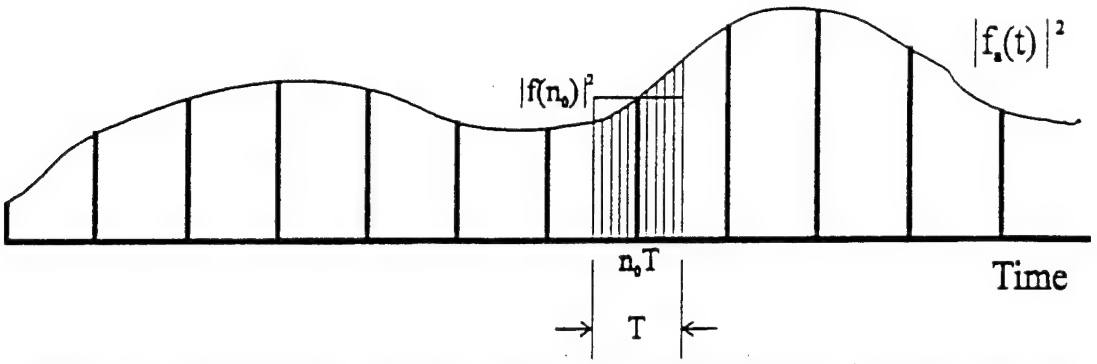



Figure 2.1. Approximation of the Energy Around $n_0T$. The Shaded Region Represents the True Energy, While the Area of the Rectangle is the Discrete Approximation

and the variance is

$$\text{Var}[a_k] = \sum_n \text{Var}[\eta(n)\Psi_{kn}] = \sum_n \text{Var}[\eta(n)]|\Psi_{kn}|^2 = \sigma^2 \sum_n |\Psi_{kn}|^2 = \sigma^2 \tag{2.13}$$

Since the operation to obtain $a_k$ from $\eta(n)$ in (2.3) is linear, $a_k$ also has a Gaussian distribution, and is $a_k \sim N(0,\sigma^2)$ [40]. Now, we need to consider the cross correlation between coefficients

$$\begin{aligned} E[a_k a_m^*] &= E\Big[\sum_n \eta(n)\Psi_{kn} \sum_p \eta(p)\Psi_{mp}^*\Big] = \\ &E\Big[\sum_n (\eta(n))^2 \Psi_{kn}\Psi_{mn}^* + \sum_n \eta(n)\Psi_{kn} \sum_{p\neq n} \eta(p)\Psi_{mp}^*\Big] = \\ &\sum_n E[(\eta(n))^2]\Psi_{kn}\Psi_{mn}^* + \sum_n E[\eta(n)]\Psi_{kn} \sum_{p\neq n} E[\eta(p)]\Psi_{mp}^* = \\ &\sum_n \sigma^2 \Psi_{kn}\Psi_{mn}^* = 0 \quad \text{when } m \neq k \end{aligned} \tag{2.14}$$

Of course, when $k = m$, $E[(a_k)^2] = \sigma^2$. So, when the basis set is orthonormal, the coefficients are uncorrelated, and so, because we are dealing with a Gaussian waveform, are statistically independent.

We are also interested in the probability distribution function (pdf) for the square of the coefficients. This can be found using the transform techniques described in [40]

$$f_Y(y) = \sum_{i=1}^{k} \frac{f_X(x_i)}{|J'(x_i)|} \tag{2.15}$$

where

| | |
|---|---|
| $f_X(\,)$ | is the pdf of the function to be transformed (Gaussian, in our case) |
| $f_Y(\,)$ | is the pdf we desire |
| $J(x)$ | is the Jacobian of the transformation, $J(x) = x^2 \equiv y$ in our case |
| $x_i$ | are the values of the variate of the original function corresponding to y |
| $y$ | is the variate of the pdf we are trying to find |
| $k$ | is the number of roots of $J(x)$ |

In our case, this leads to

$$f_{a_k^2}(y) = \frac{f_{a_k}(\sqrt{y})}{2\sqrt{y}} + \frac{f_{a_k}(\sqrt{y})}{2\sqrt{y}} = \frac{1}{\sqrt{2\pi y\sigma^2}}\exp[-y/(2\sigma^2)] \tag{2.16}$$

which, if the noise variance, $\sigma^2$, is normalized to one, becomes the well known Chi-Squared distribution with one degree of freedom.

Often, we will want to add several coefficients. The pdf of the sum of random variables is the convolution of the variables' pdfs [40]. To find this, we will first compute the characteristic function of the pdf in (2.16)

$$\begin{aligned}\Xi_{a_k^2}(\omega) &\equiv \int_{-\infty}^{\infty} f_{a_k^2}(y)\exp(j\omega y)dy = \\ &\frac{1}{\sqrt{2\pi\sigma^2}}\int_0^{\infty}\frac{1}{\sqrt{y}}\exp\left[\left(j\omega - \frac{1}{2\sigma^2}\right)y\right]dy = (1 - j2\omega\sigma^2)^{-1/2}\end{aligned} \tag{2.17}$$

The characteristic function is the complex conjugate of the continuous Fourier transform, and so we can find the characteristic function of a sum of random variables by multiplying the characteristic functions. In our case, for $\gamma$ coefficients, we will have

$$\Xi_{a_k^2 a_{k+1}^2 \cdots a_{k+\gamma-1}^2}(\omega) = (1 - j2\omega\sigma^2)^{-\gamma/2} \tag{2.18}$$

To find the pdf, we use the characteristic function's inverse transform [40]

$$\begin{aligned}f_{a_k^2 a_{k+1}^2 \cdots a_{k+\gamma-1}^2}(y) &\equiv \frac{1}{2\pi}\int_{-\infty}^{\infty}\Xi_{a_k^2 a_{k+1}^2 \cdots a_{k+\gamma-1}^2}(\omega)\exp(-j\omega y)d\omega = \\ &\frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{\exp(-j\omega y)}{(1 - j2\omega\sigma^2)^{\gamma/2}}d\omega = \\ &\frac{1}{2^{\gamma/2}\Gamma(\gamma/2)\sigma^{\gamma}}y^{\gamma/2-1}\exp\left(\frac{-y}{2\sigma^2}\right)\end{aligned} \tag{2.19}$$

Where $\Gamma(\bullet)$ is the gamma function. When $\sigma^2$ is normalized to one, (2.19) becomes the Chi-Squared function with $\gamma$ degrees of freedom.

Now, let's examine the coefficients if the waveform consists of a deterministic signal plus band limited WGN

$$f(n) = s(n) + \eta(n)$$

First we have

$$E[f(n)] = E[s(n)] + E[\eta(n)] = s(n)$$

$$Var[f(n)] = Var[s(n)] + Var[\eta(n)] = \sigma^2$$

Since the waveform variance is equal to the noise variance, from (2.13) we have

(2.20) $$Var[a_k] = \sigma^2$$

From (2.3), we have

(2.21) $$E[a_k] = \sum_n E[f(n)]\Psi_{kn} = \sum_n s(n)\Psi_{kn}$$

so

(2.22) $$a_k \sim N\left(\sum_n s(n)\Psi_{kn}, \sigma^2\right)$$

Also, from the well known relation in probability theory, we have

(2.23) $$E[a_k^2] = Var[a_k] + \{E[a_k]\}^2 = \sigma^2 + \left\{\sum_n s(n)\Psi_{kn}\right\}^2$$

From (2.13) and (2.23), we can find the ratio of energy contributed to each coefficient by the deterministic signal, to the energy contributed by the band limited WGN

(2.24) $$SNR_{a_k} = \frac{\left\{\sum_n s(n)\Psi_{kn}\right\}^2}{\sigma^2}$$

We can now examine the difference a good choice for a basis set can make on detection. To keep things straight, let's define the first basis set to be $\{\Omega\}$, and the coefficients, found from (2.3), to be $\{p\}$; and the second basis set to be $\{\Phi\}$, and its coefficients $\{q\}$. For the sake of example, let's say a signal, s(n), is spanned by K functions of the first set, and that, further, the coefficients are all equal. On the other hand, the same signal is spanned by a single function of the second set. This gives us

$$(2.25)\quad SNR_{p_k} = \frac{\left\{\sum_n s(n)\Omega_{kn}\right\}^2}{\sigma^2} = \frac{\left\{\sum_n \frac{s(n)}{K}\Phi_{kn}\right\}^2}{\sigma^2} = \left(\frac{1}{K}\right)^2 \frac{\left\{\sum_n s(n)\Phi_{kn}\right\}^2}{\sigma^2} = \left(\frac{1}{K}\right)^2 SNR_q$$

Thus we have an improvement by a squared factor in the SNR of each the coefficients by using the second basis set [4].

We now want to find the pdf for the coefficient squared when the waveform consists of a deterministic signal plus band limited WGN. First, for notational convenience, let

$$m_k \equiv E[a_k] = \sum_n s(n)\Psi_{kn}$$

then the pdf for $a_k$ is

$$(2.26)\qquad f_{a_k}(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-(x-m_k)^2/2\sigma^2\right]$$

As in Equation (2.15), we have a Jacobian, $J(x) = x^2 = y$. As with (2.16), we have

$$(2.27)\qquad f_{a_k^2}(y) = \frac{f_{a_k}(x_1)}{J'(x_1)} + \frac{f_{a_k}(x_2)}{J'(x_2)}$$

where

$$\sqrt{y} \equiv x_1 \equiv -x_2$$

so we have

$$f_{a_k^2}(y) = \frac{1}{2\sqrt{2\pi\sigma^2 y}}\left\{\exp\left[-(x_1^2 - 2x_1m_k + m_k^2)/2\sigma^2\right] + \exp\left[-(x_2^2 - 2x_2m_k + m_k^2)/2\sigma^2\right]\right\} =$$

$$\frac{1}{2\sqrt{2\pi\sigma^2 y}}\left\{\exp\left[-(x_1^2 + m_k^2)/2\sigma^2\right]\exp\left[x_1m_k/\sigma^2\right] + \right.$$

$$\left. \exp\left[-(x_1^2 + m_k^2)/2\sigma^2\right]\exp\left[-x_1m_k/\sigma^2\right]\right\} =$$

$$\frac{1}{\sqrt{2\pi\sigma^2 y}}\exp\left[-(x_1^2+m_k^2)/2\sigma^2\right]\left\{\frac{\exp\left[x_1 m_k/\sigma^2\right]+\exp\left[-x_1 m_k/\sigma^2\right]}{2}\right\}=$$

$$\frac{1}{\sqrt{2\pi\sigma^2 y}}\exp\left[-(y+m_k^2)/2\sigma^2\right]\cosh\left[m_k\sqrt{y}/\sigma^2\right]$$

(2.28)

which, if $\sigma^2$ is normalized to one, is the Chi-Squared distribution with one degree of freedom and with a non-centrality parameter of $m_k^2$. Using the characteristic function on (2.28), we can find the pdf for a sum of coefficients when both signal and noise energy are present, and it turns out to be [41]

$$f_{a_k^2 a_{k+1}^2 \cdots a_{k+\gamma-1}^2}(y) = \frac{1}{2\sigma^2}\left(\frac{y}{\lambda}\right)^{\frac{(\gamma/2-1)}{2}}\exp\left(-\frac{\lambda+y}{2\sigma^2}\right)I_{\gamma/2-1}\left(\frac{\sqrt{\lambda y}}{\sigma^2}\right) \tag{2.29}$$

where $I_{\gamma/2-1}(\bullet)$ is the modified Bessel function of the first kind and order $\gamma/2-1$, and

$$\lambda = \sum_{i=k}^{k+\gamma-1} m_i^2$$

and, therefore, represents the signal energy contributed to the coefficients. When $\sigma^2$ is normalized to one, (2.29) is the Chi-Squared distribution with $\gamma$ degrees of freedom and with a non-centrality parameter of $\lambda$.

By comparing the functions described by (2.19) and (2.29), we now can see how a detector can decide whether a signal is present. Figure 2.2 shows these functions for the particular case where we are adding two squared coefficients, and $\sigma^2$ is equal to one. $f_n(Y)$ is the pdf when noise only is present, and $f_s(Y)$ is the pdf when noise and a signal are both present. As the figure indicates, we can set a threshold, Th, and compare the sum of the squared coefficients against it, deciding a signal is present if the observed number exceeds the threshold. Using this threshold detection method, we can compute the probability of detection, given that a signal is present, by integrating under $f_s(Y)$ to the right of Th (the shaded region labeled $Q_d$ in the figure). Likewise, our probability of false alarm when no signal is present is computed similarly, and is indicated by $Q_{fa}$.

To summarize our main points so far: 1) To detect energy concentrations due to a signal we want to decompose the waveform into a basis set such that the signal energy is contained in as few terms as possible. Our ability to do this will, of course, depend on both our knowledge of the signal, and on our ability to find a basis set that spans the signal space and is orthonormal. 2) If no signal is present, the squared coefficients will have a Chi-Squared pdf. If it is present, the squared coefficients containing signal energy will have a Chi-Squared pdf with a non-centrality parameter. Therefore, threshold detection is possible.

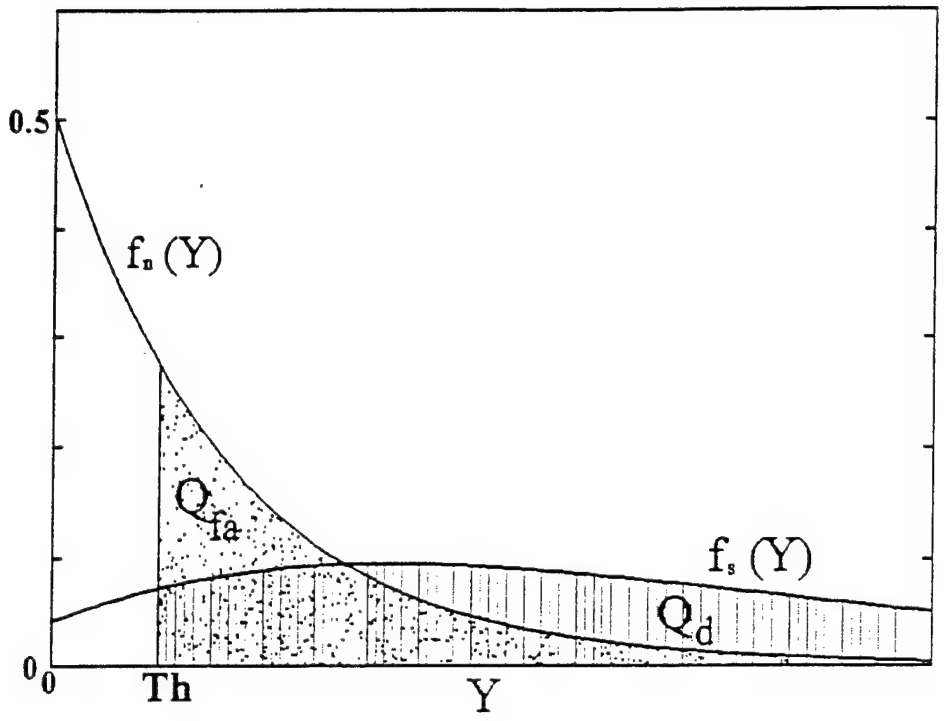



Figure 2.2. Chi-Square Probability Distribution Functions

**Common Examples of Basis Sets**

We now present some common examples of basis sets which should be familiar to most readers. This is done both to reinforce the concepts discussed above and to introduce new issues necessary in the discussion of Wavelets that follows. For the remainder of this paper, we will assume the input waveforms are purely real.

**Matched Filter.** A matched filter can be thought of, mathematically, as a basis set containing a single function. In this case, we are trying to detect a signal, and we know everything about it (shape, arrival time, etc.) except whether it is present. This is the case, for example, at the receiver on a digital communications link, where matched filtering is widely used. The basis function is selected to be the shape of the signal, so it spans the signal space, and, because there is only one function, it is orthogonal by definition. (With the proper scaling, of course, it becomes orthonormal.) For our discussion, let's assume orthonormality, which implies the signal's energy will be unity.

From (2.3), we see the contribution to the coefficient from the signal is

$$a = \sum_n s(n)s(n) = 1 \tag{2.30}$$

and the SNR, from (2.24), is

$$SNR = \frac{\left\{\sum_n s(n)s(n)\right\}^2}{\sigma^2} = \frac{1}{\sigma^2} \tag{2.31}$$

which is, of course, all of the signal's energy over the noise energy. This is the best we can hope to do.

This can be extended to the detection of multiple signals simply by requiring the signals be mutually orthonormal, and expanding the basis set appropriately. In this way, more information can be sent down the link, with the receiver selecting the signal by finding the largest coefficient.

**The Discrete Fourier Transform (DFT).** This is a basis set familiar to most Electrical Engineers. It is

$$\Psi^*_{kn} = \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}} \tag{2.32}$$

where

$n \in [0, N - 1]$ is the time index
$k \in [0, N/2 - 1]$ is the unique frequency index (although we only restrict $k \in$ integers, see below)

and leads to the DFT pair

$$\begin{aligned} a_k &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n) e^{-j\frac{2\pi}{N}kn} \\ f(n) &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{j\frac{2\pi}{N}kn} \end{aligned} \tag{2.33}$$

A somewhat more common form, in engineering texts, replaces $1/\sqrt{N}$ with $1/N$ in the first equation, and with 1 in the second. This is simply a matter of scaling, and orthogonality of the basis set is maintained.

The coefficients, with this basis set, represent the frequency components of the waveform. The index k is not strictly limited to the range [0, N/2 - 1], but there are only N/2 unique basis functions, due to the cyclical nature of the complex exponential

$$e^{j\frac{2\pi}{N}0n} = e^0 = e^{j\frac{2\pi}{N}Nn} = e^{j2\pi n} \tag{2.34}$$

and the fact that

$$e^{j\frac{2\pi}{N}nk} = -e^{j\frac{2\pi}{N}n\frac{k}{2}} = -e^{j\frac{\pi}{N}nk} \tag{2.35}$$

The value k/N is often referred to as the digital (linear) frequency [32]. We see there are N/2 unique digital frequencies in the range [0, 1/2 - 1/N]. We show this in Figure 2.3, where the horizontal axis represents time, the vertical axis represents frequency, and each rectangular "tile" represents, approximately, the location in the time-frequency plane of the energy included in each coefficient.

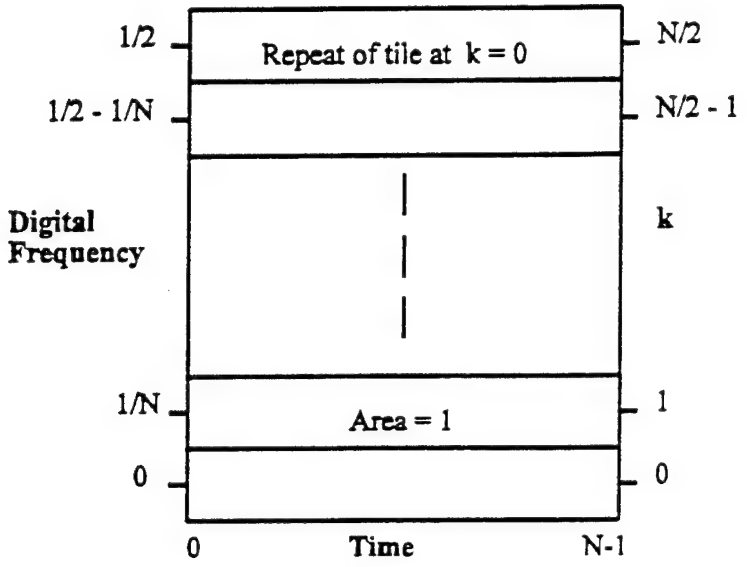



Figure 2.3. Time Frequency Diagram For the Discrete Fourier Transform

There is a fundamentally important concept to note in Figure 2.3. Observe if we have more samples to analyze (larger N), the maximum frequency remains the same, but the resolution in the frequency domain increases. The area of each tile, however, remains fixed. This limit in our ability to resolve the time frequency plane of a signal is fundamental for all orthonormal linear transform techniques. Another thing to note: if N approaches infinity, the number of tiles will approach infinity, their frequency dimension will approach zero, and it would be mathematically correct to replace the summation in the second equation of (2.33) with an integral. The frequency domain would, effectively, become continuous.

Ideally, for our purposes, all of the energy in each tile would contribute equally to that tile's coefficient, and $|a_k|^2$ will represent the total energy in the tile. Unfortunately, this is not possible. To explore this, let's find the frequency response of each basis function. Consider a finite time waveform of length N, consisting of a single frequency k/N. Mathematically, we can view this as an infinite time waveform ($n \in [-\infty, \infty]$), multiplied by a "rectangular window": a discrete function that consists of values of one for $n \in [0, N-1]$, and zero otherwise. Because we consider these waveforms

to be of infinite length, we can, as stated above, consider the frequency domain to be continuous. Now, a property of the DFT is [32]

(2.36) $$DFT[g(n)w(n)] = DFT[g(n)] \otimes DFT[w(n)]$$

where

| | |
|---|---|
| DFT[ ] | represents the DFT of the argument |
| g(n) | is our, now infinite time, input function |
| w(n) | is our window |
| $\otimes$ | represents convolution |

The DFT of g(n) is an impulse at k/N. The DFT of the rectangular window, on the other hand, is the sinc function (2.7). Actually, due to the window's discrete nature, the DFT repeats every m(N + k/N) for m $\in$ integers, but as stated above, we only consider k from [0, N/2 - 1]. Convolving this with g(n) will give us a sinc centered at k/N. Considering several adjacent basis functions, the true tiling of the frequency domain is shown in Figure 2.4.

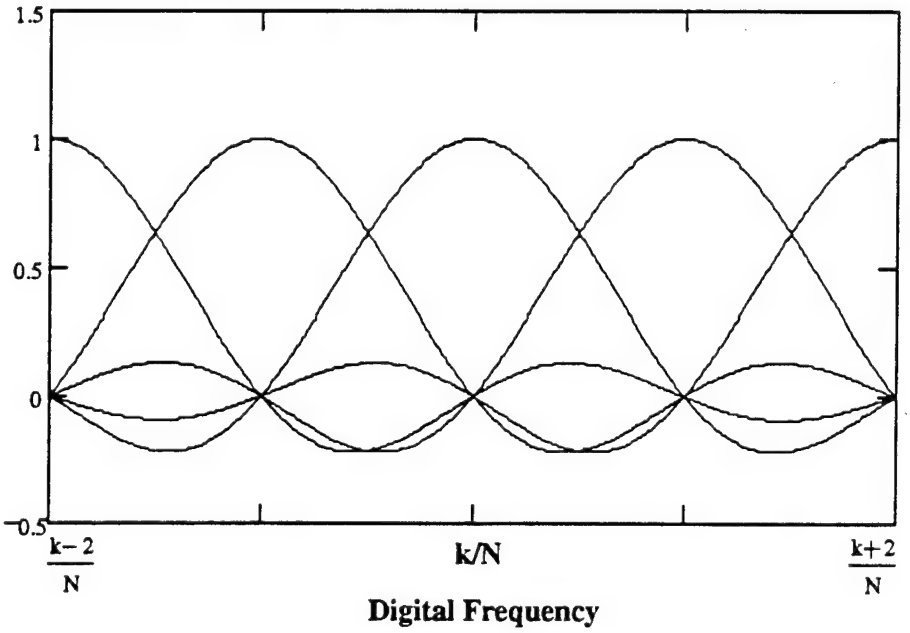



Figure 2.4. True Coverage of the Frequency Domain With a Rectangular Window in the Time Domain

To find the values of the coefficients for any arbitrary waveform, we could multiply in the time domain, as indicated by (2.3), which translates to a (digital) convolution in the frequency domain. However, each shift performed in the convolution will be for $k = mN$ for m $\in$ integers, and, as above, we are only concerned with k from [0, N/2 - 1], so we only need consider a single shift.

The result is that we can find the values of the coefficients by considering the frequency response of an infinite length input waveform, and multiplying it with the sinc curves in Figure 2.4.

To see the result of this, consider an input waveform consisting of a single digital frequency of k/N. In the frequency domain, of course, it would be an impulse at k/N. As Figure 2.4 shows, all of the basis functions, except one, have nulls at this frequency. Hence, all of the coefficients, except $a_k$, will be zero. On the other hand, consider an input waveform consisting of a single digital frequency of $k/N + \varepsilon$, where $\varepsilon$ is a small number (less than 1/2N). In this case, $a_k$ will be the largest coefficient, but will not be as large as in the previous case. Also, adjacent coefficients will no longer be zero. This is sometimes called "leakage" in the literature [32] [33]. Of course, Equation (2.4) will still be satisfied.

Leakage can be mitigated by replacing the implicit rectangular window described above with truncated time domain windows of other shapes [33]. (To retain the orthonormality of Equation (2.2), and the result of Equation (2.4), the basis functions must be rescaled.) Many window shapes have been devised, each with certain advantages. For our purposes, we would like as much of the energy as possible that would appear in a tile to be represented by that tile's coefficient. There's an obvious trade off here: improving the frequency characteristics will make the time characteristics less desirable. One possible compromise is to use a (truncated) Gaussian shaped window. The DFT of a Gaussian shape in the time domain will yield a Gaussian shape in the frequency domain.

This gives us the tiling diagram as shown in Figure 2.5, where the oval shapes represent contours [24]. Because the window is (necessarily) truncated in time, there will still be sidelobes in the frequency domain, but they will be much smaller than they would for the rectangular window. Thus, most of the energy in a tile will be reflected by the proper coefficient. Unfortunately, energy near the center of the tile will contribute more than energy near the edge. One way around this problem is to increase the number of basis functions--forsaking orthonormality--and overlapping the tiles. Of course, this increases the computation work involved, as well as making the evaluation more complex.

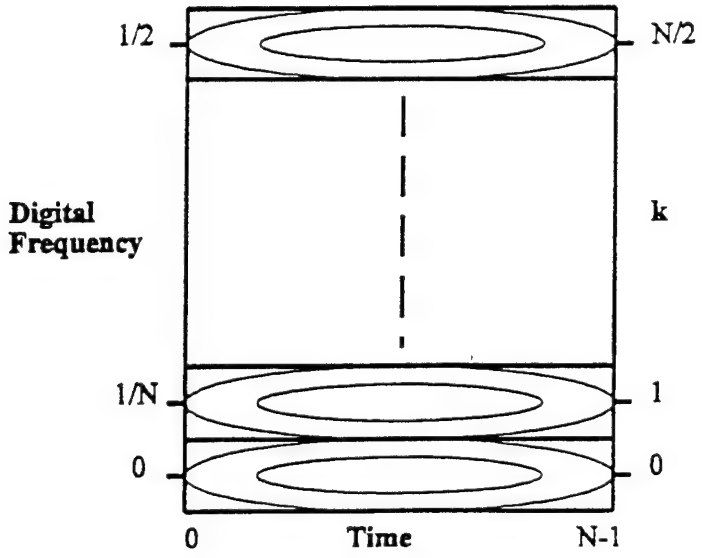



Figure 2.5. Time Frequency Diagram For the DFT With Gaussian Window

For the detection of signals in noise, the DFT works best for signals that are stationary over the evaluation time. The DFT is second nature to most Electrical Engineers. So much so, that when a signal is designed specifically to be covert, frequency hopping and pseudo-noise spreading techniques are often used to "hide" the signal in noise when the waveform is evaluated with the DFT.

**The Short Time Fourier Transform (STFT).** For signals that are stationary (or nearly stationary) for short periods of time, an obvious modification of the DFT is to break up the waveform into shorter sequences, and perform a DFT on these sequences. In this case, we take the index k, from Equation (2.32), break it into two indices, say i and m, and our basis set becomes

$$\Psi^*_{imn} = \frac{e^{j\frac{2\pi}{N}i(mN+n)}}{\sqrt{N}} \tag{2.37}$$

where

| | |
|---|---|
| $i \in [0, N/2 - 1]$ | frequency index |
| $n \in [0, N - 1]$ | time index |
| $m \in$ integers | time shift |

This gives us the tiling diagram shown in Figure 2.6.

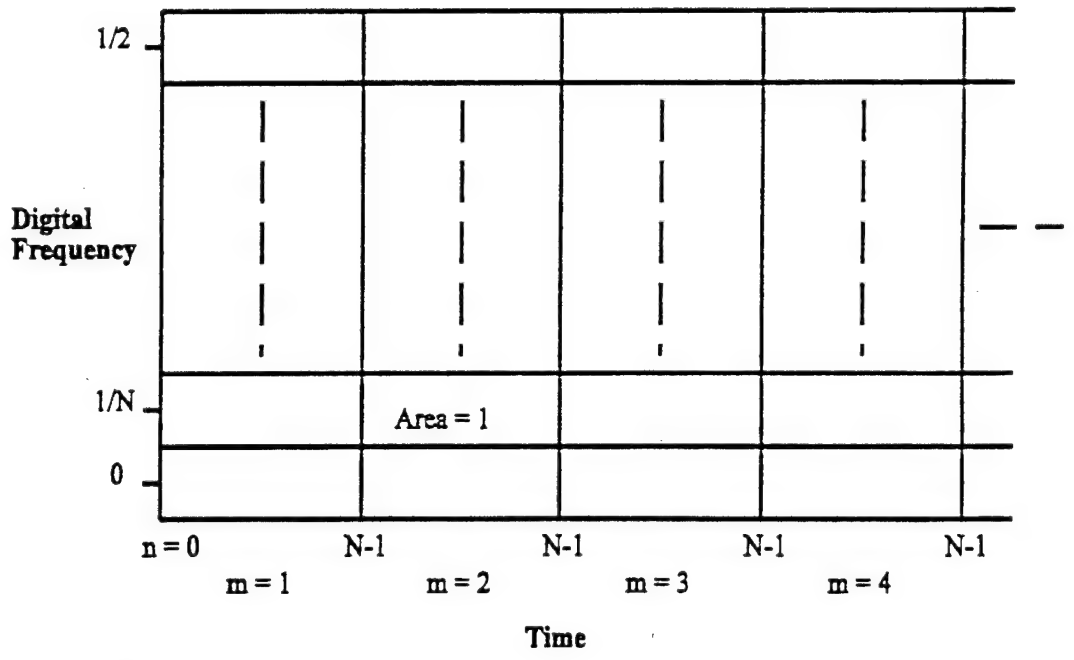



Figure 2.6. Time Frequency Diagram For the Short Time Fourier Transform

This set is orthonormal, since columns of tiles do not overlap in time. As with the DFT, Gaussian or other windows may be used to improve the representation. Also as before, overlapping tiles may be used, at the expense of orthonormality, and with an increase in the required number of computations.

## Wavelet Transforms

In the last section, we saw that a time frequency analysis of an input waveform using orthonormal basis sets can be represented as a tiling of the time frequency plane. The square of each coefficient roughly represents the energy in a particular tile.

One drawback to the STFT is that all of the tiles must have the same dimensions. Many signals (including, in particular, naturally occurring signals) have a "constant Q" characteristic: their duration in time decreases proportionally as their frequency is increased. Wavelet Transforms tile the time frequency plane in this manner, and are, therefore, a useful class of basis functions for detecting these types of signals. In addition, the rules for the construction of filter banks to accomplish Wavelet decomposition also lead to methods for accomplishing a more-or-less arbitrary tiling of the time frequency plane.

Since many readers may be unfamiliar with Wavelet Transforms, we will discuss their development in some detail in this section. (Most of the material presented here comes from [44] and [12].) As will be seen, once the mathematical framework is laid down, relatively straight-forward rules for implementation can be derived. In the next section we will extend these rules to allow for arbitrary tiling.

### Mathematical Framework

We have already discussed the properties basis functions must have for detection: they must span our signal space and be orthonormal. We will only have to evaluate the functions at discrete points, but, for purposes of mathematical development, continuous functions will be considered.

The general Wavelet basis set is

$$\Psi_{ab}(t) = \frac{1}{\sqrt{a}}\Psi\left(\frac{t-b}{a}\right) \tag{2.38}$$

where

| | |
|---|---|
| $\Psi(\,)$ | is the "mother" wavelet |
| b | is the translation variable, b is real |
| a | is the dilation variable, a is real and greater than zero |
| t | is the (continuous) independent variable--always time in this paper |

The fundamental idea is to use a mother wavelet that is zero except for a finite time interval, and that oscillates in that interval. We then obtain our basis functions by shifting the wavelet along the time interval (changing b), and by shrinking the wavelet (changing a), to decrease the period of the oscillations.

We have imposed two conditions on the mother wavelet: 1) It must be zero, except for a finite interval. In the mathematical literature, it is said to have "compact support" [10] [44]. To meet this, we require

$$\int_{-\infty}^{\infty} |\Psi(t)| dt < \infty \tag{2.39}$$

2) It must oscillate. We ensure this by imposing the "admissibility condition" [12]

$$\int_{-\infty}^{\infty} \frac{\left|\hat{\Psi}(\omega)\right|^2}{\omega} d\omega < \infty \tag{2.40}$$

where $\hat{\Psi}(\omega)$ is the Fourier Transform of the mother wavelet. With this requirement, we see the mean must be zero, and, therefore, the mother wavelet will have to oscillate.

Since we will only evaluate the wavelet at discrete points, we can impose

$$\begin{aligned} a &= 2^m \\ b &= n2^m \end{aligned} \qquad \text{for } m,n \in \text{integers} \tag{2.41}$$

on (2.38) to get

$$\Psi_{mn}(t) = \frac{1}{\sqrt{2^m}} \Psi\left(\frac{t}{2^m} - n\right) \tag{2.42}$$

This gives us a binary dilation and a dyadic translation. (Other possibilities exist, but this is presented most often in the literature [10] [12] [24] [44].) The tiling of the time frequency diagram, for this case, is shown in Figure 2.7.

To summarize: we need a basis set meeting (2.42) that spans our space of signals, has orthonormal functions, and has a mother wavelet with compact support and zero mean. We can develop functions that meet these conditions through multi-resolution analysis [12].

In multi-resolution analysis we deal with function spaces. Specifically, we define a set of subspaces of functions such that

$$\begin{aligned} &\ldots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \ldots \subset L^2(\Re) \\ &\bigcap V_m = \{0\} \quad \text{for } m \in \text{integers} \\ &f(\bullet) \in V_m \Leftrightarrow f(2\bullet) \in V_{m-1} \\ &f(\bullet) \in V_m \Rightarrow f(\bullet - 2^m n) \in V_m \end{aligned} \tag{2.43}$$

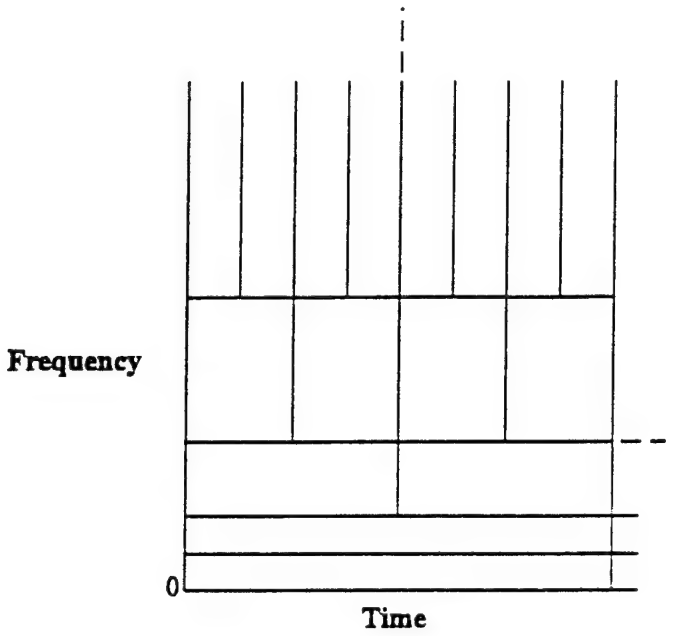



Figure 2.7. Time Frequency Diagram For the Wavelet Transform

where the last two lines deal with dilation and translation of a function and the corresponding relationship of the subspaces. $L^2(\Re)$ is the set of functions that are square integrable (which, necessarily, includes all of our real world waveforms).

Now, let there be a function, $\Phi(t) \in V_0$, and

$$\Phi_{mn}(t) = \frac{1}{\sqrt{2^m}}\Phi\left(\frac{t}{2^m} - n\right) \tag{2.44}$$

such that $\Phi_{0n}(t)$ is orthonormal for all $n \in$ integers,

$$\int \Phi_{0n}(t)\,\Phi_{0k}(t)\,dt = \begin{cases} 1 & k = n \\ 0 & k \neq n \end{cases} \tag{2.45}$$

and $\Phi_{mn}(t)$ spans $V_m$. $\Phi(t)$ is called the scaling function and defines the subspaces. An example is shown in Figure 2.8.

Consider a continuous function, f(t), that we wish to decompose, using multiresolution analysis. We need its projection onto the different subspaces. This is the same as saying we want to approximate it by linear combinations of $\Phi_{mn}(t)$ for each $V_m$. To do this, denote the projection operator as $P_m$, and we'll have

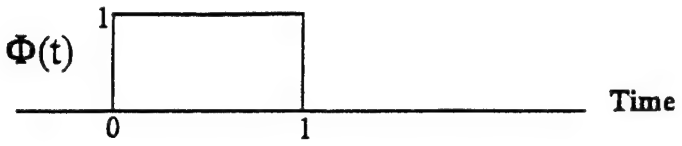



Figure 2.8a. An Example of a Scaling Function

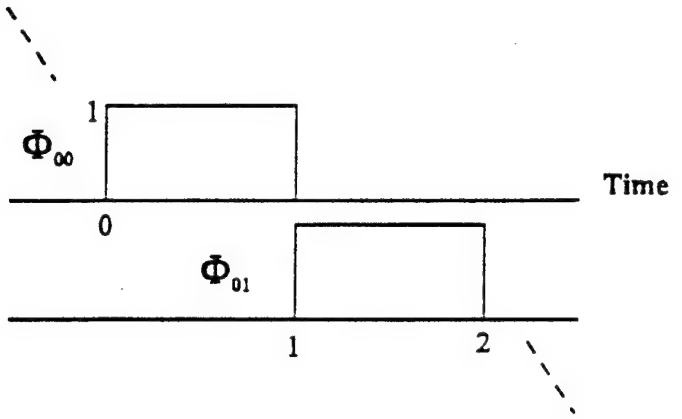



Figure 2.8b. Translated Versions of the Scaling Function.
All Linear Combinations of These Are Functions in $V_0$

$$\begin{aligned} &P_0 f(t) \in V_0 \\ &P_{-1} f(t) \in V_{-1} \\ &P_{-2} f(t) \in V_{-2} \\ &\text{etc.} \end{aligned} \tag{2.46}$$

where each projection with a smaller subscript will be more accurate (it will have finer resolution). Also, denote the difference between projections with another operator, $Q_m$, such that

$$Q_m f(t) = P_{m-1} f(t) - P_m f(t) \tag{2.47}$$

It is easy to see, then that f(t) is the sum of the difference projections

$$f(t) = \sum_m Q_m f(t) \tag{2.48}$$

Now we define another set of subspaces, $W_m$ such that

$$V_{m-1} = V_m \oplus W_m \qquad W_m \perp V_m \tag{2.49}$$

where $\oplus$ denotes direct sum: the combination of two subspaces. From this, we can see

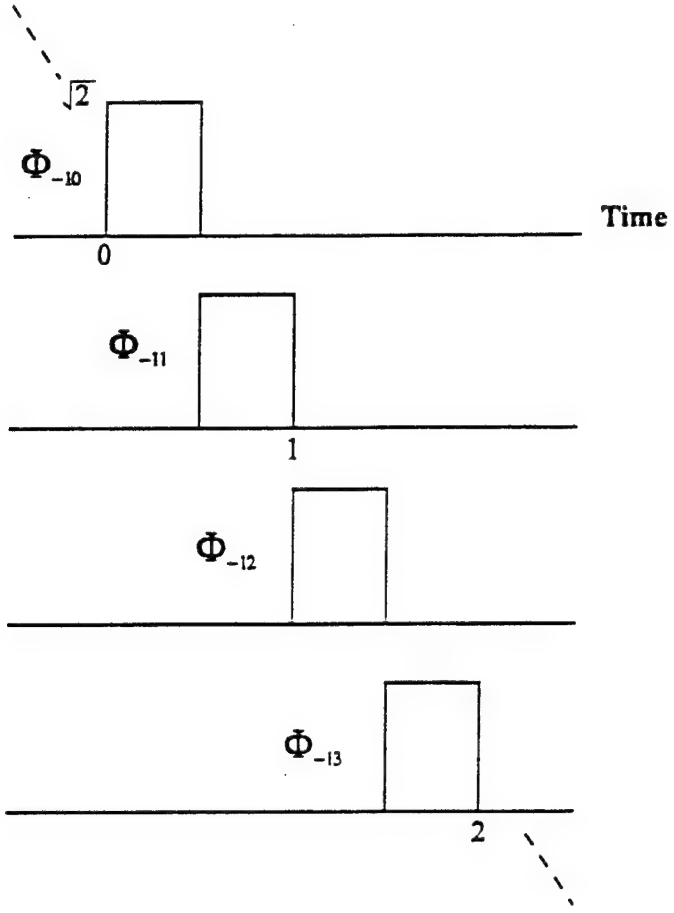



Figure 2.8c. Dilated and Translated Versions of the Scaling Function.
All Linear Combinations of These Are Functions in $V_{-1}$

$$W_m \perp W_n \quad \text{for } m \neq n$$

$$\bigoplus_m W_m = L^2(\Re) \tag{2.50}$$

and

$$Q_m f(t) \in W_m \tag{2.51}$$

Now, since $\Phi(t) \in V_0 \subset V_{-1}$ which is spanned by $\Phi(2t - n)$, there must be

$$\Phi(t) = \sqrt{2} \sum_n h(n)\Phi(2t - n) \tag{2.52}$$

for some {h(n)}. This is called the "dilation equation". (The $\sqrt{2}$ is included for later convenience.) It can then be shown [12]

$$\Psi(t) = \sqrt{2}\sum_{n}(-1)^{n}h(n+1)\Phi(2t+n) \tag{2.53}$$

and

$$\Psi_{mn}(t) = \frac{1}{\sqrt{2^{m}}}\Psi\left(\frac{t}{2^{m}}-n\right) \tag{2.54}$$

can be constructed, where $\Psi_{mn}(t)$ spans $W_m$.

It is evident from (2.48), (2.50) and (2.51) that (2.54) meets our requirements for a basis set, as long as a scaling function can be found such that the dilation equation gives us a sequence {h(n)} with only a finite number of non-zero elements. This last condition comes from (2.53) and our requirement that the mother wavelet have compact support.

One example where this is true is given by the scaling function in Figure 2.8. From examination, and the dilation equation, we see

$$h(0) = h(1) = 1/\sqrt{2}$$

and our mother wavelet will be as shown in Figure 2.9. This is the Haar basis set.

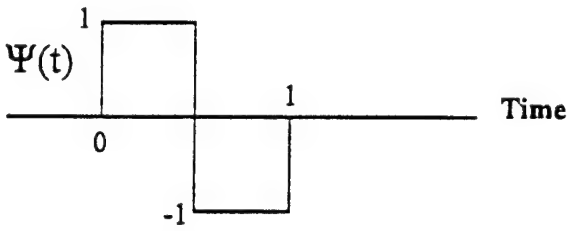



Figure 2.9. Mother Wavelet For the Haar Basis Set

In the next sub-section, we will find it convenient to have an equation for the coefficients in terms of the scaling function. So, before proceeding, we derive it: Let

$$t = \frac{x}{2}-r$$

for $r \in$ integers. Then (2.52) becomes

$$\Phi\left(\frac{x}{2} - r\right) = \sqrt{2}\sum_n h(n)\Phi[x - (n + 2r)] \tag{2.55}$$

and, applying (2.44)

$$\Phi_{1r}(x) = \sum_n h(n)\Phi_{0(n+2r)}(x) \tag{2.56}$$

From (2.45) we can solve (2.56) in a manner similar to (2.3)

$$\begin{aligned} \int \Phi_{1r}(x)\Phi_{0m}(x)dx &= \int \sum_n h(n)\Phi_{0(n+2r)}(x)\Phi_{0m}(x)dx = \\ &\sum_n h(n)\int \Phi_{0(n+2r)}(x)\Phi_{0m}(x)dx = \\ &h(m-2r) \end{aligned} \tag{2.57}$$

This is the equation we desired.

**Wavelet Filter Bank**

We now will use multi-resolution analysis, and the results above, to design a filter bank with the tree structure shown in Figure 2.10. We denote the discrete input waveform as the sequence $\{c_n^0\}$, and the outputs of each branch as shown in the figure. Since each branch of the tree decimates by 2, each sequence will have half as many elements as the sequence preceding it. Our goal will be to design the filters to give us the coverage shown in Figure 2.11, where each output sequence, $\{d_n^k\}$, represents the coefficients for a row of tiles.

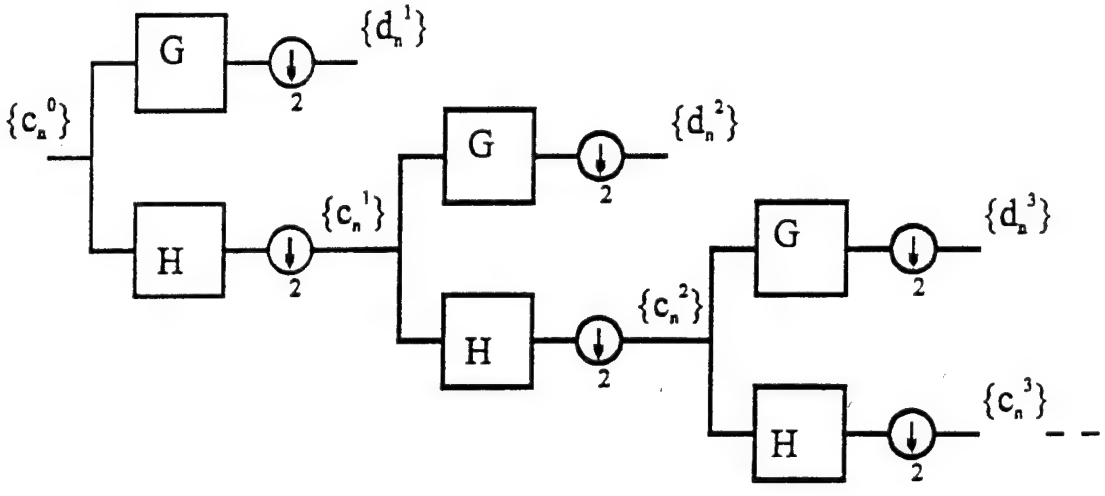



Figure 2.10. Wavelet Filter Bank

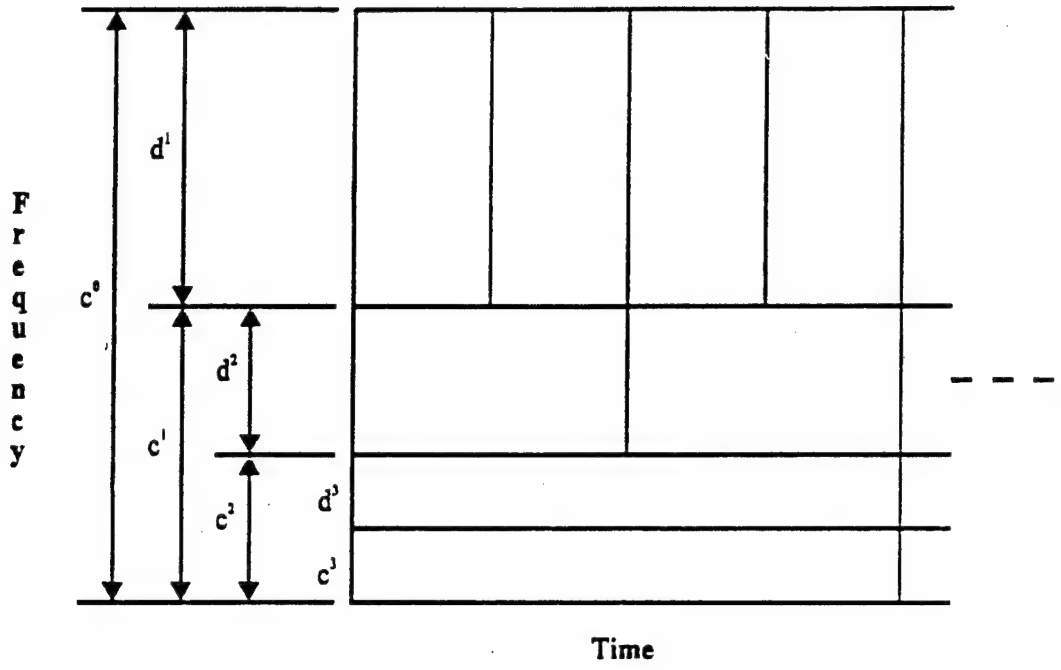



Figure 2.11. Time Frequency Diagram For the Wavelet Filter Bank

We begin by assuming we have an input sequence, and have selected our sub-spaces, $V_m$, and an appropriate scaling function. We then construct a continuous time function

$$f(t) = \sum_n c_n^0 \Phi_{0n}(t) \tag{2.58}$$

where $f(t) \in V_0$ so $P_0 f(t) = f(t)$. Since $V_0 = V_1 \oplus W_1$, $P_1 f(t) \in V_1$, and $Q_1 f(t) \in W_1$, we can write $f(t) = P_1 f(t) + Q_1 f(t)$. Now, by expanding

$$P_1 f(t) = \sum_k c_k^1 \Phi_{1k}(t) \tag{2.59}$$

and

$$Q_1 f(t) = \sum_k d_k^1 \Psi_{1k}(t) \tag{2.60}$$

we will have the sequences $\{c_n^1\}$ and $\{d_n^1\}$ shown in Figure 2.10. Since $P_1 f(t)$ has half of the resolution of $f(t)$, $\{c_n^1\}$ will cover the lower half of the frequency band, as required in Figure 2.11. Since this is so, the "difference", $\{d_n^1\}$, must cover the upper half.

To solve (2.59) and (2.60) for the sequences, we start by noting the requirement that $\Phi_{0k}(t)$ is orthonormal for all integer values for k, and Equation (2.44) will then make $\Phi_{1k}(t)$ orthonormal

for all integer values for k. $\Psi_{1k}(t)$, on the other hand, is orthonormal from our initial requirement for orthonormal basis functions and our subsequent development.

As with the solution of (2.56), we solve (2.59) by multiplying each side by $\Phi_{1n}(t)$ and integrating over time. Since $\Phi_{1k}(t)$ is orthonormal, we have

$$c_k^1 = \int P_1 f(t) \Phi_{1k}(t) dt \tag{2.61}$$

Since $P_1 f(t) \in V_1 \subset V_0$, $\Phi_{1k}(t)$ spans $V_1$, and $f(t) \in V_0$, we can substitute f(t) for $P_1 f(t)$ in (2.61), and get

$$c_k^1 = \int f(t) \Phi_{1k}(t) dt \tag{2.62}$$

Substituting (2.58) into (2.62), and rearranging, will then give us

$$c_k^1 = \sum_n c_n^0 \int \Phi_{0n}(t) \Phi_{1k}(t) dt \tag{2.63}$$

The integral is what we found in (2.57), so we have

$$c_k^1 = \sum_n h(n-2k) c_n^0 \tag{2.64}$$

With a similar development we can solve for $\{d_k^1\}$ from (2.60). Define

$$g(n-2r) = \int \Phi_{0n}(t) \Psi_{1r}(t) dt \tag{2.65}$$

Then we find

$$d_k^1 = \sum_n g(n-2k) c_n^0 \tag{2.66}$$

Equation (2.64) may be written as a matrix equation

$$\begin{bmatrix} \vdots \\ c_{-1}^1 \\ c_0^1 \\ c_1^1 \\ c_2^1 \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \ddots & & \\ \cdots & h(2) & h(3) & h(4) & h(5) & \cdots \\ & h(0) & h(1) & h(2) & h(3) & \\ & h(-2) & h(-1) & h(0) & h(1) & \\ & h(-4) & h(-3) & h(-2) & h(-1) & \\ & & & \ddots & & \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ c_{-1}^0 \\ c_0^0 \\ c_1^0 \\ c_2^0 \\ c_3^0 \\ c_4^0 \\ \vdots \\ \vdots \end{bmatrix}$$

(2.67)

which we will denote as $\mathbf{c_1 = H\ c_0}$. Notice the rows of H shift to the right by 2 as they progress down the matrix. This can be implemented as a Finite Impulse Response (FIR) filter with a decimation by 2, just as we desired. Notice also, that when the scaling function has compact support, so most of the elements of {h(n)} are zero, H will be a sparse matrix.

In a similar way, (2.66) may be written in matrix form as

(2.68) $$\mathbf{d_1 = G\ c_0}$$

where G will also have rows shifting to the right by 2. (2.67) and (2.68) give us a design for the first layer of the tree filter in Figure 2.10. Since, throughout this development, we have not specified the frequency band, but only relationships relative to the initial bandwidth, it is easy to see lower branches of the filter will have the same structure, and will be implemented with the same H and G.

So we see that if we have an appropriate scaling function: one that meets the dilation equation, (2.52), with a finite number of non-zero coefficients, as well as meeting the requirement for generating orthonormal translates, we can compute the elements of H and G, and implement the Wavelet Transform as a filter bank. Unfortunately, finding scaling functions is difficult. In the next sub-section, however, we will develop some rules which will allow the generation of H and G without an explicit scaling function.

**Rules For the Coefficients**

**Rule 1.** First, we impose

(2.69) $$\sum_n |h(n)| \neq \infty$$
$$\sum_n |g(n)| \neq \infty$$

to meet our requirement for compact support for the mother wavelet. (The first part of (2.69) comes from (2.53). The second part of (2.69) will follow from the first part and the equation for g(n) which we will develop below.) Actually, since we only consider FIR filters, our requirement will be even stronger: we will only allow a finite number of non-zero coefficients.

**Rule 2.** Here we derive a rule for the coefficients from the orthonormality requirement for the scaling function, (2.45). From that requirement, and (2.44), we get

$$\int \Phi(t+j)\Phi(t+k)\,dt = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases} \tag{2.70}$$

Now, substituting in (2.52) we get

$$2\int \Big(\sum_n h(n)\Phi(2t+2j-n)\Big)\Big(\sum_m h(m)\Phi(2t+2k-m)\Big)dt =$$

$$\text{let } n' = n-2j \text{ and } m' = m-2k$$

$$2\int \sum_{n'} h(n'+2j)\Phi(2t-n')\sum_{m'} h(m'+2k)\Phi(2t-m')dt = \tag{2.71}$$

$$2\sum_{n'}\sum_{m'} h(n'+2j)h(m'+2k)\int \Phi(2t-n')\Phi(2t-m')dt = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$$

But by substituting $x = 2t$ into the equation under the integral, we notice it is 1/2 times (2.70), so, when n' = m', we have

$$\sum_{n'} h(n'+2j)h(n'+2k) = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases} \tag{2.72}$$

which is the rule we desire. In matrix notation, it becomes

$$\mathbf{H}\,\mathbf{H}^T = \mathbf{I} \tag{2.73}$$

which says the rows of **H** are orthonormal. One of the consequences of this is that there must be an even number of non-zero coefficients [44].

**Rule 3.** For the next two rules, we wish to relate **G** to **H**. Substituting (2.44) and (2.54) into our definition for g(n), (2.65), we get

$$g(n) = \frac{1}{\sqrt{2}}\int \Psi\left(\frac{x}{2}\right)\Phi(x-n)\,dx \tag{2.74}$$

Substituting (2.53) then gives us

$$g(n) = \int \sum_m (-1)^m h(m+1)\Phi(x+m)\Phi(x-n)dx =$$

$$\sum_m (-1)^m h(m+1) \int \Phi(x+m)\Phi(x-n)dx = \tag{2.75}$$

$$(-1)^n h(1-n)$$

where the last step follows from the orthonormality relation of (2.70). This gives us the following matrix

$$\mathbf{G} = \begin{bmatrix} \ddots & & & & & & & & & \\ \cdots & h(L-1) & -h(L-2) & \cdots & \cdots & h(1) & -h(0) & 0 & 0 & \cdots \\ & 0 & 0 & h(L-1) & -h(L-2) & \cdots & \cdots & h(1) & -h(0) & \\ & & & & & \ddots & & & & \end{bmatrix} \tag{2.76}$$

where there are L non-zero coefficients in both G and H. There is something important to note from (2.76): because G is of infinite extent, we can shift all of the rows to the right or left by multiples of two, and the only result, in (2.66), will be a shift in the time index on $\mathbf{d_k}$. In [24] and [44] an alternative definition

$$g(n) = (-1)^n h(L-1-n) \tag{2.77}$$

is used. This gives the same frequency response for the filters, permits a reconstruction of the original sequence (which is the intention of the two papers), and makes some manipulation easier. When constructing the time frequency diagram, however, particularly when arbitrary tiling is used, it is important to keep the time shift in mind. For the rest of this section, we will use (2.75). Later, when we describe Wavelet filters useful for detection, we will see certain shifts of the rows in (2.76) will be necessary.

We see G is just H with the row elements reversed and some minus signs added. From Rule 2, we get the following, which we call Rule 3:

$$\mathbf{G}\,\mathbf{G}^T = \mathbf{I} \tag{2.78}$$

(It is easy to see the minus signs cancel each other.)

**Rule 4.** Rearranging (2.75) to find h in terms of g, we get

$$h(m) = -(-1)^{-m} g(m+1) \tag{2.79}$$

Substituting this into (2.72) for one of the h's, we get

$$\sum_{n'} h(n'+2j)[-(-1)^{-(n'+2k)} g(n'+2k+1)] =$$

$$-\sum_{n'} h(n'+2j)(-1)^{n'} g(n'+2k+1) =$$

(2.80) let $2k+1=2p \quad p \in$ integers

so $k = p-(1/2)$

$$-\sum_{n'} (-1)^{n'} h(n'+2j) g(n'+2p) = \begin{cases} 1 & p-(1/2) = j \\ 0 & p-(1/2) \neq j \end{cases}$$

but, since p and j are both integers, (2.80) must always equal zero. So, Rule 4 is

(2.81) $$\mathbf{H}\,\mathbf{G}^{\mathbf{T}} = \mathbf{0}$$

where **0** is a matrix with all elements equal to zero.

**Some Consequences of Rules 2 - 4.** We saw in the first section of this chapter that an orthonormal basis set implies that the transform is invertible (i.e., the signal can be reconstructed from the decomposed sequences). To see how this applies here, consider Figure 2.12, giving us the following equation

(2.82) $$\mathbf{c_0} = \mathbf{G^T G}\, \mathbf{c_0} + \mathbf{H^T H}\, \mathbf{c_0} = (\mathbf{H^T H} + \mathbf{G^T G})\, \mathbf{c_0}$$

or

(2.83) $$\mathbf{H^T H} + \mathbf{G^T G} = \mathbf{I}$$

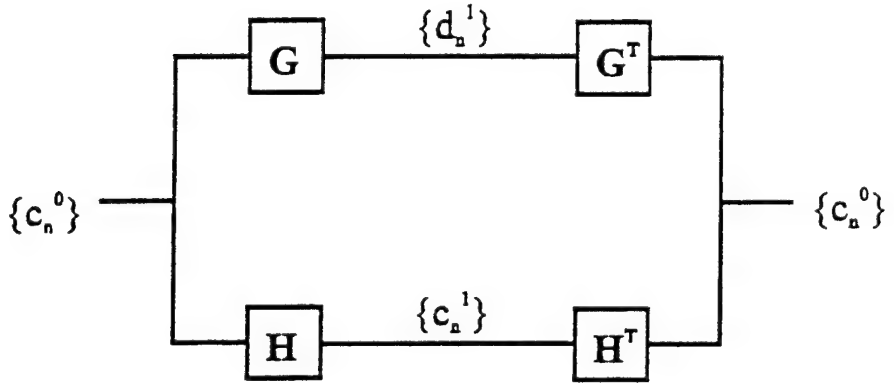



Figure 2.12. Decomposition and Reconstruction of a Sequence. Blocks H and G include a decimation by 2 at the filter outputs. Blocks $\mathbf{H^T}$ and $\mathbf{G^T}$ include an expansion by 2 (the insertion of a zero valued sample between each input sample) preceding each filter.

We now show the orthonormality rules for the Wavelet filters make (2.83) true. (2.83) in a non-matrix form can be written as

$$\sum_{i\,\text{even}}\left[h(i)^2 + g(i)^2\right] = 1 \tag{2.84}$$

$$\sum_{i\,\text{odd}}\left[h(i)^2 + g(i)^2\right] = 1 \tag{2.85}$$

$$\sum_{i}\left[h(i)h(i+n) + g(i)g(i+n)\right] = 0 \quad \text{for } \forall n \neq 0 \tag{2.86}$$

so we must derive these from Rules 2 - 4. Now, from (2.73) we get

$$\sum_{i} h(i)^2 = 1 \tag{2.87}$$

and using this and (2.79) yields

$$\begin{aligned} &\sum_{i\,\text{even}} h(i)^2 + \sum_{i\,\text{odd}} h(i)^2 = \sum_{i\,\text{even}} h(i)^2 + \sum_{i\,\text{odd}}\left[-(-1)^i g(i+1)\right]^2 = \\ &\sum_{i\,\text{even}}\left[h(i)^2 + g(i)^2\right] = 1 \end{aligned} \tag{2.88}$$

which is (2.84). Of course, (2.85) can be derived similarly. To derive (2.86) from the rules above, note that from (2.73) and (2.78) we get

$$\sum_{i} h(i)h(i+n) = 0 \quad \text{and} \quad \sum_{i} g(i)g(i+n) = 0 \quad \text{for } \forall \text{ even } n \neq 0 \tag{2.89}$$

so

$$\sum_{i}\left[h(i)h(i+n) + g(i)g(i+n)\right] = 0 \quad \text{for } \forall \text{ even } n \neq 0 \tag{2.90}$$

which is part of (2.86). Now, assuming n is odd, and using (2.75) we see

$$\begin{aligned} &\sum_{i}\left[h(i)h(i+n) + g(i)g(i+n)\right] = \\ &\sum_{i}\left[h(i)h(i+n)\right] + \sum_{i}\left[(-1)^i h(1-i)(-1)^{i+n} h(1-i-n)\right] = \\ &\qquad\qquad \text{let } 1-i-n = k \\ &\sum_{i}\left[h(i)h(i+n)\right] - \sum_{k}\left[h(k)h(k+n)\right] = 0 \quad \text{for } \forall \text{ odd } n \end{aligned} \tag{2.91}$$

which is the other part of (2.86).

The ability to reconstruct the signal is often called the "Perfect Reconstruction" (PR) property in the literature. Several recent papers, [1] [7] [8] [9] [24] [26] [27] [44] for example, have used Wavelets in sub-band coding, where the necessity of PR is obvious. (It should be noted in passing that orthogonality implies PR, but not the reverse. In particular, there are a class of "biorthogonal" Wavelet filters which satisfy the PR property and also have a linear phase response [44]. These, apparently, are of no interest for us, since for energy detection we are not particularly interested in the phase, and we do require orthogonality for equation (2.4) to be satisfied.)

A property we do need, however, is the "Power Complementary" (PC) property. As with PR, we will first state the result, and then show how it is implied by orthonormal Wavelets. Actually, since we have already shown orthonormality leads to PR, it will be convenient to use (2.84) - (2.86).

First, the Fourier Transform of filters H and G (before decimation) are

$$H(\omega) = \sum_n h(n)e^{-j\omega n} \quad \text{and} \quad G(\omega) = \sum_n g(n)e^{-j\omega n} \tag{2.92}$$

and the PC property is

$$|H(\omega)|^2 + |G(\omega)|^2 = c \tag{2.93}$$

where c is a constant. The intuitive description of (2.93) is that, no matter what the frequency components of an input sequence, all of its energy will be at the outputs of the two filters. (No energy will be lost between the pass bands.) Combining (2.92) with the left side of (2.93), we have

$$|H(\omega)|^2 + |G(\omega)|^2 =$$

$$\left[\sum_n h(n)e^{-j\omega n}\right]\left[\sum_k h(k)e^{-j\omega k}\right] + \left[\sum_n g(n)e^{-j\omega n}\right]\left[\sum_k g(k)e^{-j\omega k}\right] =$$

$$\sum_n\sum_k h(n)h(k)e^{-j\omega(n-k)} + \sum_n\sum_k g(n)g(k)e^{-j\omega(n-k)} =$$

$$\text{let } i = k-n$$

$$\sum_n\sum_i \left[h(n)h(i+n) + g(n)g(i+n)\right]e^{j\omega i} =$$

$$\sum_i e^{j\omega i}\left(\sum_n \left[h(n)h(i+n) + g(n)g(i+n)\right]\right) \tag{2.94}$$

From (2.86) we can see that the part of (2.94) in parenthesis on the last line will be zero when i is not zero. When i is zero, we get

$$|H(\omega)|^2 + |G(\omega)|^2 = \sum_n \left[h(n)^2 + g(n)^2\right] =$$

(2.95)

$$\sum_{n\ \text{even}} \left[h(n)^2 + g(n)^2\right] + \sum_{n\ \text{odd}} \left[h(n)^2 + g(n)^2\right] = 2$$

where the last part comes from (2.84) and (2.85). As we see, the constant in (2.93) is two.

**Rule 5.** Rules 2-4 ensure we decompose the original sequence into two orthogonal sequences. Now, we need to ensure $\{d_n^{\,1}\}$ contains the high frequency components and $\{c_n^{\,1}\}$ the low frequency components. We first consider the Fourier Transform of G given in (2.92). To make G a high pass filter, we want it to block direct current (DC), so the first half of Rule 5 is

(2.96) $$G(0) = \sum_n g(n) = 0$$

On the other hand, to analyze **H**, we consider an input sequence with zero frequency (pure DC). We arbitrarily scale the input sequence so it is $\{c_n^{\,0}\} = \{1\}$, where $\{1\}$ is a sequence of ones. We then want $\{d_n^{\,1}\} = \{0\}$ and $\{c_n^{\,1}\} = \{\gamma\}$, where $\{0\}$ is a sequence of zeros, and $\{\gamma\}$ is a sequence of arbitrary, single valued, non-zero, elements. Looking at Figure 2.12, we see we want

(2.97) $$\boldsymbol{\gamma} = \mathbf{H}\,\mathbf{1}$$

where

$\boldsymbol{\gamma}$ is a column vector with all elements equal to $\gamma$
$\mathbf{1}$ is a column vector with all elements equal to 1.

(2.97) gives us

(2.98) $$\gamma = \sum_n h(n)$$

which, when we find $\gamma$, will be the second half of our Rule 5. To do this, we look back at Figure 2.12 and note

(2.99) $$\mathbf{H}^{\mathbf{T}}\boldsymbol{\gamma} = \mathbf{1}$$

or

$$\begin{bmatrix} & \ddots & & \\ h(0) & 0 & 0 & \\ h(1) & 0 & 0 & \\ \vdots & h(0) & 0 & \\ h(L-1) & h(1) & 0 & \\ \cdots\ 0 & \vdots & h(0) & \cdots \\ 0 & h(L-1) & h(1) & \\ 0 & 0 & \vdots & \\ 0 & 0 & h(L-1) & \\ & \ddots & & \end{bmatrix} \begin{bmatrix} \vdots \\ \gamma \\ \gamma \\ \gamma \\ \gamma \\ \gamma \\ \gamma \\ \gamma \\ \gamma \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \end{bmatrix}$$

(2.100)

Examining (2.100), we see we can extract the following pair of equations

$$\begin{aligned} \left[\sum_j h(2j)\right]\gamma &= 1 \\ \left[\sum_k h(2k+1)\right]\gamma &= 1 \end{aligned} \tag{2.101}$$

or

$$\begin{aligned} \sum_j h(2j) &= \frac{1}{\gamma} \\ \sum_k h(2k+1) &= \frac{1}{\gamma} \end{aligned} \tag{2.102}$$

which we can manipulate as follows

$$\begin{aligned} \sum_p h(2p)\sum_j h(2j) &= \frac{1}{\gamma^2} \\ \sum_q h(2q+1)\sum_k h(2k+1) &= \frac{1}{\gamma^2} \end{aligned} \tag{2.103}$$

but, from (2.72) we can see the left hand sides of (2.103) will be zero when $p \neq j$ and $q \neq k$. So we have

$$\sum_j [h(2j)]^2 = \frac{1}{\gamma^2}$$

(2.104)

$$\sum_k [h(2k+1)]^2 = \frac{1}{\gamma^2}$$

Combining these, we get

(2.105) $$\sum_j [h(2j)]^2 + \sum_k [h(2k+1)]^2 = \frac{2}{\gamma^2} = \sum_n [h(n)]^2 = 1$$

where the last step comes from (2.87). So we have

$$\frac{2}{\gamma^2} = 1$$

(2.106)

$$\gamma = \sqrt{2}$$

Finally, Rule 5 in its entirety is

$$\sum_n g(n) = 0$$

(2.107)

$$\sum_n h(n) = \sqrt{2}$$

**The Regularity Criterion.** In addition to the five rules just presented, references [12] [24] [44] consider one more requirement. Since we have not considered the effects of cascading the filter, it is possible to follow our rules, and come up with a set of coefficients that result in a non-continuous scaling function. The "regularity criterion" handles this by requiring the scaling function (and, perhaps, its first, second, etc. derivatives) be continuous. This is not critical for our purposes here, since we are concerned strictly with discrete time signals.

For the purposes of detection, alternative criteria will be more useful. Specifically, we are interested in filters that will collect as much of the signal energy within a time frequency tile as possible, while rejecting most of the energy outside the tile. Discussion of techniques for computing the values of coefficients for filters of this type will be discussed in Chapter III.

## Arbitrary Tiling

In the last section, we saw that by cascading filters and filtering the low pass component of the previous output we achieved a tiling with finer frequency resolution at lower frequencies. In many cases concerning detection, however, this is not desired. Many man made signals, for example, will have a constant bandwidth over a wide range of center frequencies. (American television channels, as a specific example, all have a 6 Mhz bandwidth, but range from 54 to 770 Mhz--over an order of magnitude! [21].) Instead, we desire an arbitrary tiling, adjusted to meet specific requirements for the type of signal we are trying to detect. As it happens, we can modify the filter bank presented in the last section to do this, and we discuss some of the possibilities here. Much of this material comes from [24].

Consider the cascading filter diagram in Figure 2.13 where, instead of filtering the low pass output of each stage, we filter the high pass output. Since our derivation of the filter coefficients in the last section only concerned relative frequencies, we again split the input sequence at each stage into high frequency and low frequency orthogonal sequences, and we have the tiling diagram shown in Figure 2.14, sometimes referred to as "Wavelet Packet Tiling" [24].

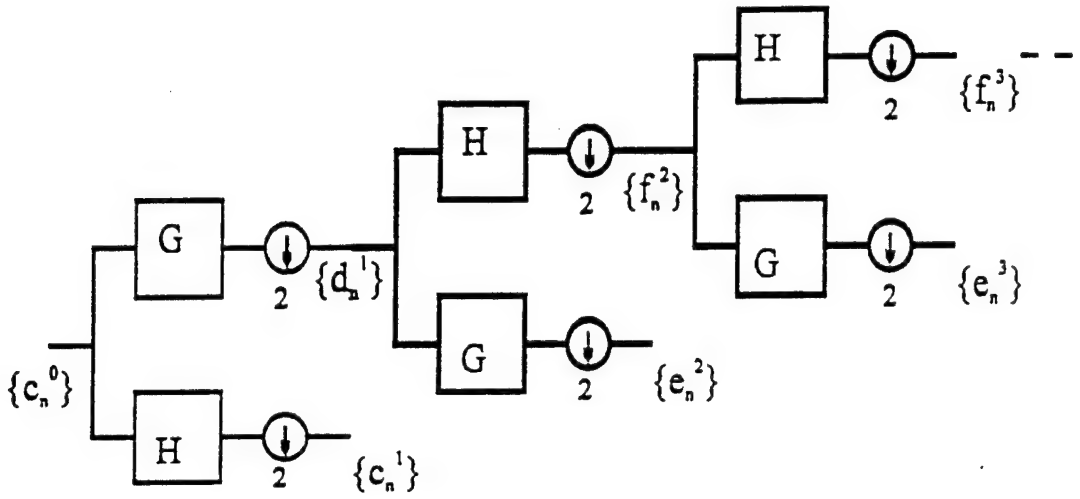



Figure 2.13. Wavelet Packet Filter Bank

Notice the second and third layer seem to be flipped in Figure 2.13 (the figure is drawn so the output sequence at the top of the drawing contains the higher frequency components of the input sequence). This will be important for us a little later. To see why they are flipped, consider the aliased frequency spectrum of the filters, shown in Figure 2.15. We see the output from the G filter in the first layer contains the highest frequency component of the original sequence, but shifted, so it is actually the DC component of the output of G. The result is that the output of G is frequency reversed, much like the lower sideband of a single sideband communications system. Of course, a similar filtering further down the cascade will "unflip" the signal. It turns out there is a simple rule to keep our output straight, which we will discuss in a moment.

Combining Wavelet tiling and Wavelet packet tiling will allow us to create another tiling scheme. This is shown in Figure 2.16, and gives us the tiling diagram of Figure 2.17. The construction rule for Figure 2.16, in order to keep the higher frequency outputs of each branch above the lower frequency outputs, is to count the number of G filters up to the branch. If the number is even, the next G filter will output the high frequencies. If odd, the next H filter will output the high frequencies.

Figure 2.17 looks similar to the STFT tiling diagram. It is, except the tiles are overlapped in time, as well as frequency, and each tile in the figure has only half of the area of an STFT tile, giving us better resolution.

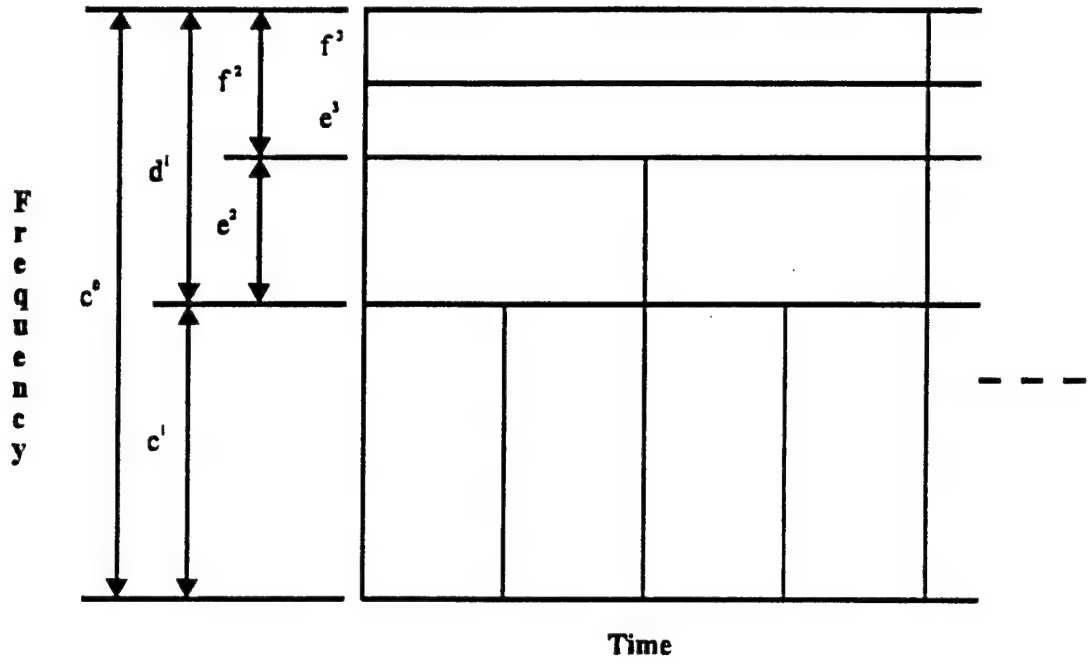



Figure 2.14. Time Frequency Diagram For the Wavelet Packet Filter Bank

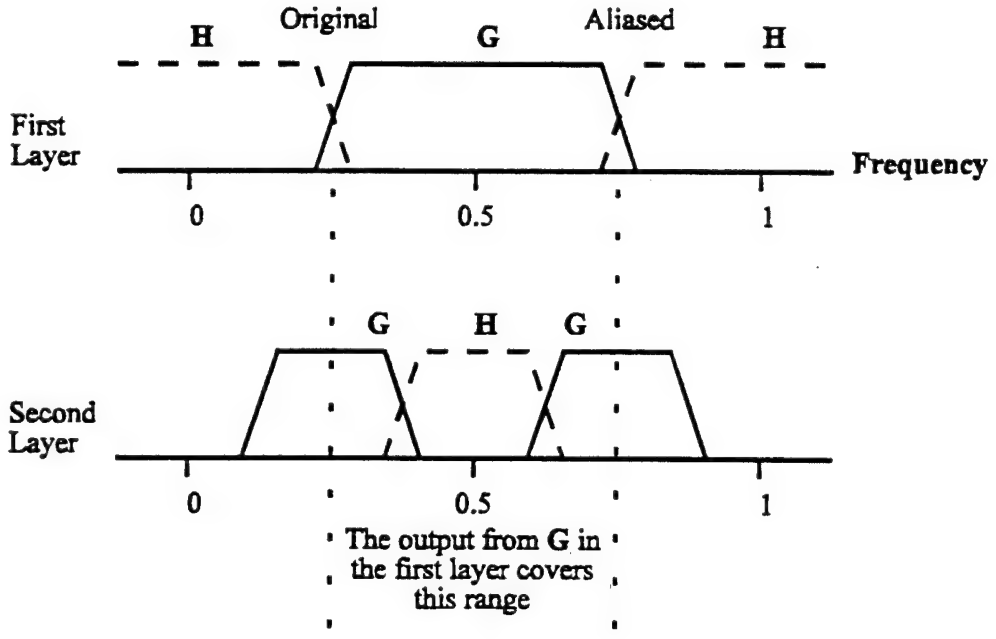



Figure 2.15. Response of Filters in Figure 2.13

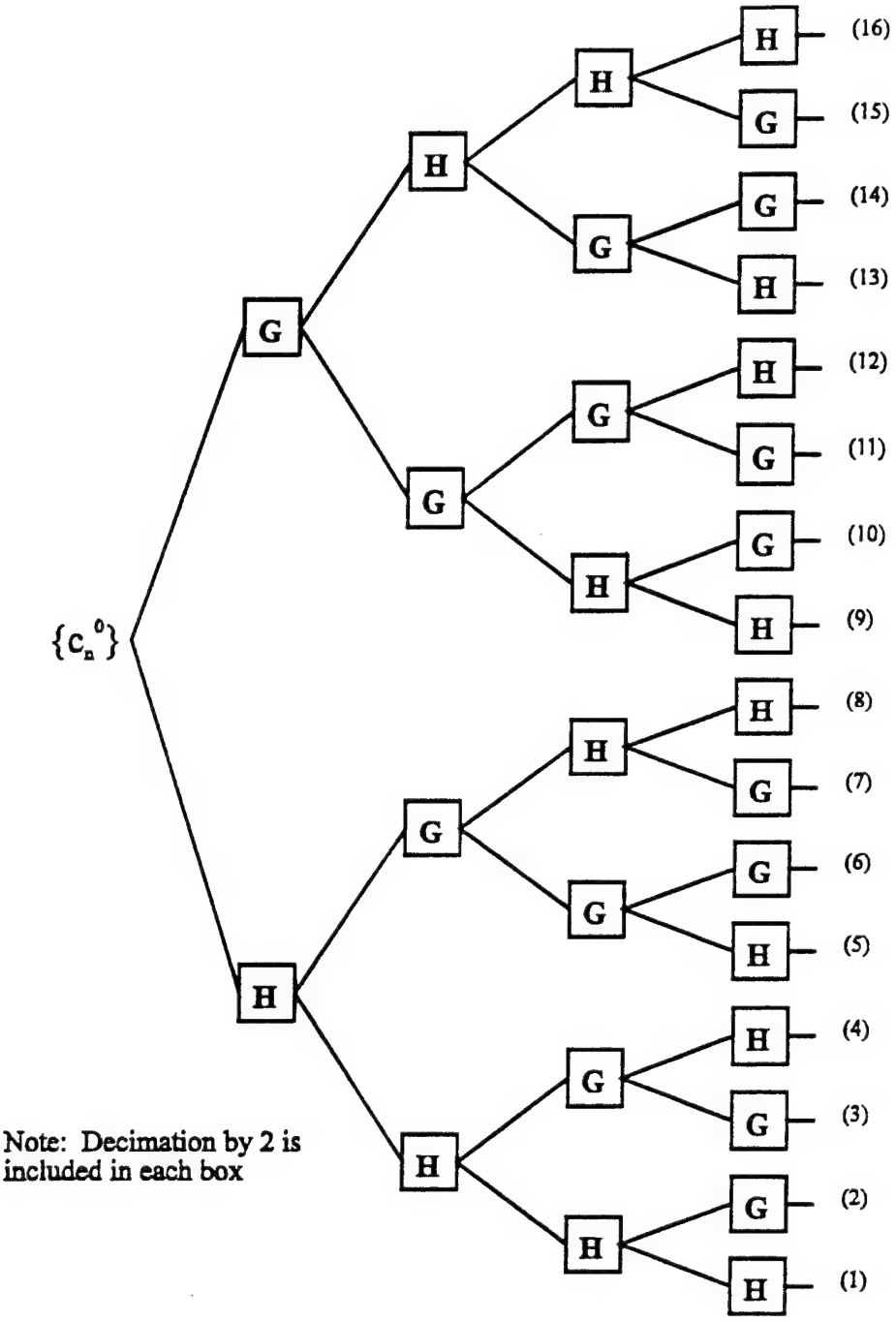



Figure 2.16. Combining the Wavelet and Wavelet Packet Filter Banks

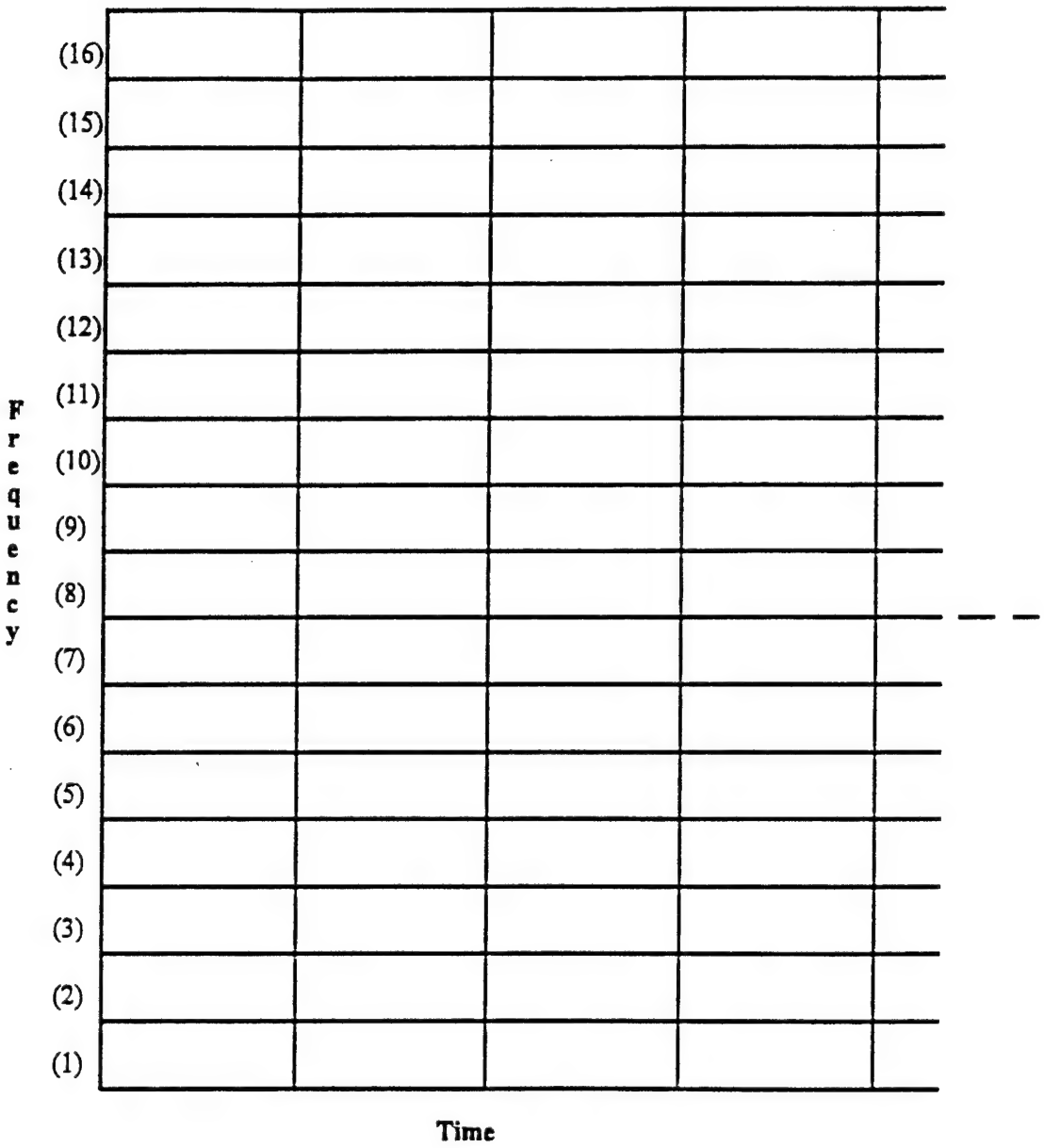



Figure 2.17. Time Frequency Diagram For the Filter Bank in Figure 2.16

To see why these tiles should cover half of the area, consider the portion of the time frequency plane with digital linear frequencies ranging from -0.5 to 0.5. A real signal has equal positive and negative components, so its energy distribution will appear as a mirror image, reflected around the zero frequency line.

Now, for the sake of example, consider a 32 sample long segment of the signal and take the DFT. We will obtain 32 coefficients, of which the 16 tiling the plane from frequencies 0 to 0.5 will be unique. Therefore, 16 tiles will cover an area of (32)(0.5) = 16, so each tile will have an area of one.

FIR filters, on the other hand, tile the plane differently. The filter passband in the positive frequency region will have a corresponding mirrored passband in the negative region. In Figure 2.15, for example, the H filter in layer 2 with a passband centered at 0.5 will also have a passband at -0.5. Therefore, each filter will pass both the positive and negative frequency components of the signal. To see that the tiles in Figure 2.17 have an area of 0.5, consider the 32 sample long segment again. In this case, at any layer of the filter bank the total output from all of the filters will contain 32 elements, so we have 32 tiles, each one unique. Since we are covering the same portion of the time frequency plane as above, each tile must have half of the area of the STFT.

It is easy to see that by expanding only certain branches of the filter tree we can achieve a variety of tiling diagrams. Furthermore, it is also possible to modify the tree with time, making available even more possibilities.

There is another possible way of using a fully developed filter tree like the one in Figure 2.16. A closely related problem to detection is feature extraction. For example, when looking for frequency hopped (FH) signals, particularly in a location where many FH transmitters are present, it would be advantageous for a detector to estimate the length of time between each hop (the dwell time), and the frequency spread. Since sequences from higher up the filter tree have a greater resolution in time, they may be useful for estimating the dwell time, while sequences down the tree, having a better frequency resolution, could be used for estimation along that dimension. Using this fact, it should be possible for a detector to simultaneously detect and classify multiple transmitters.

## Conclusion

To summarize: for detection, we need to select an orthonormal set of basis functions that will include as much of the signal energy in as few terms as possible. The "best" basis set, then, depends on what, and how much, is known about the signal to be detected. Once the basis set is selected, and the waveform is decomposed, the presence or absence of a signal may be decided via threshold detection.

For stationary signals, the best choice is generally the DFT. For non-stationary signals, the STFT, Wavelets, and variations on Wavelet filtering are all possibilities. In all of these sets, a major question is the (3 dimensional) shape of the tile and its overlap, in both the time and frequency domains. This question, for the fully developed filter tree like the one shown in Figure 2.16, is the subject of the next chapter.

## III. Filter Coefficients

### Introduction

In this chapter we build on the mathematical foundation laid out previously for orthogonal Wavelet filters, and find filters with properties desirable for use in energy detection. Specifically, we will first consider our basic requirements, and the characteristics of filters as a part of the filter tree of Figure (2.16) (which, for the rest of this paper, we will refer to as the quadrature mirror filter (QMF) bank tree). Several filters widely discussed in the literature will then be analyzed with these requirements in mind. Finally, we will see how to develop the orthogonal Wavelet filters that best meet our needs.

### Basic Requirements

Obviously, any filter we consider must meet (or at least approximately meet) the five rules laid out in the last chapter for orthogonal Wavelet filters. The first of these (the finite sum of the absolute values of the coefficients) we will meet by only considering FIR filters. The other rules will be discussed as necessary later in the chapter.

Since we assume that, in general, the cell size, hop synchronization, and channelization of the spread spectrum signals to be detected are unknown, we will not attempt to try approximating a matched filter. (Which would otherwise, as discussed in Chapter II, yield the best results.) Instead we wish to find filters closely approximating the ideal tiling characteristics shown in Figure 2.17. That is, filters passing as much signal energy as possible from within a tile, while rejecting as much as possible from outside it.

Consider first the frequency response characteristics of the prototype (low pass, or "H") filter: It should meet the specifications as shown in Figure 3.1. We would like $\delta_1$ to be as small as possible to minimize the variance in the amount of energy passed due to frequency shifts in the signal, as long as the energy remains in the bandwidth of the tile. We also want $\delta_2$ to be as small as possible to minimize the amount of energy passed outside the tile. These requirements are really the same: by meeting the PC requirement of (2.95), the frequency response of the G filter will be a mirror image of the H filter's, and a small ripple in the passband will result in little energy passed in the stop band. Also due to (2.95), the H and G filters will, between them, pass all of the signal energy. However, to minimize the amount of energy divided between them (which makes detection less likely), we would like as small of a transition region, $\Delta\omega$, as possible.

In the time domain, we specify our requirement a bit differently from most of the literature. There, the total number of FIR filter coefficients is usually considered, and minimized to reduce the amount of out of time (and, therefore, out of tile) energy collected. Really, though, only two of the coefficients collect signal from within the time dimension of the tile. All of the rest contribute to out of tile energy. What we want then, is to declare two of the coefficients to be "main taps" and to make these as large as possible. All of the others, in turn, should be as small as possible. One intuitive way to think of this is to consider the two main taps as the tile energy collectors, and the others as necessary to keep the filter's frequency response under control. We also want the two main taps to be as equal to each other as possible. In this way, energy at the input in each of the two samples falling in the tile's length (in time) will be equally represented.

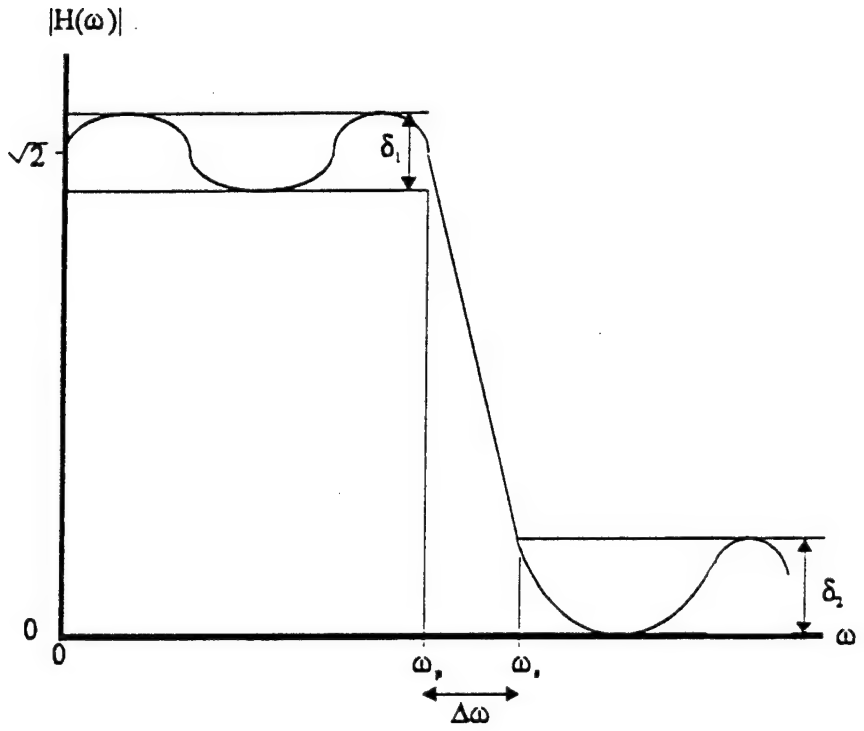



Figure 3.1. Desired Frequency Response

Any two adjacent taps in our prototype filter can be made the main taps. When constructing the G filter, however, it is important to synchronize the outputs so both H and G filters cover tiles beginning and ending at the same time. Fortunately, this is easy to do simply by adding an even number of zero coefficients (pure delay) to the filters, as shown in Figure 3.2. Referring back to Chapter II, this amounts to a shift in G's coefficients relative to the H's which may be, in general, different from either (2.75) or (2.77).

There are also several filter characteristics frequently discussed in the Wavelet literature that we do not particularly care about in our application. These are mentioned here mainly for the benefit of the readers familiar with the literature, and to show our requirements, while in many ways similar to those of others working with Wavelets, are in other respects quite different.

For example, Daubechies [12] has shown it is impossible for a FIR filter, except for the Haar filter, to meet the orthogonality requirements and have symmetric coefficients. (The proof depends on having a finite number of coefficients. Later, we will see how we can cheat by truncating an infinite length filter, at the expense of perfect orthonormality.) Symmetric coefficients in a FIR filter give it a linear phase response--a characteristic that is often desirable. Since we are looking for energy, however, the filter's phase characteristics are of no importance. (We assume we don't know the phase of the signal to be detected anyway. If we did, we would also have to know other signal characteristics, and we could consider matched filters and/or coherent detection.)

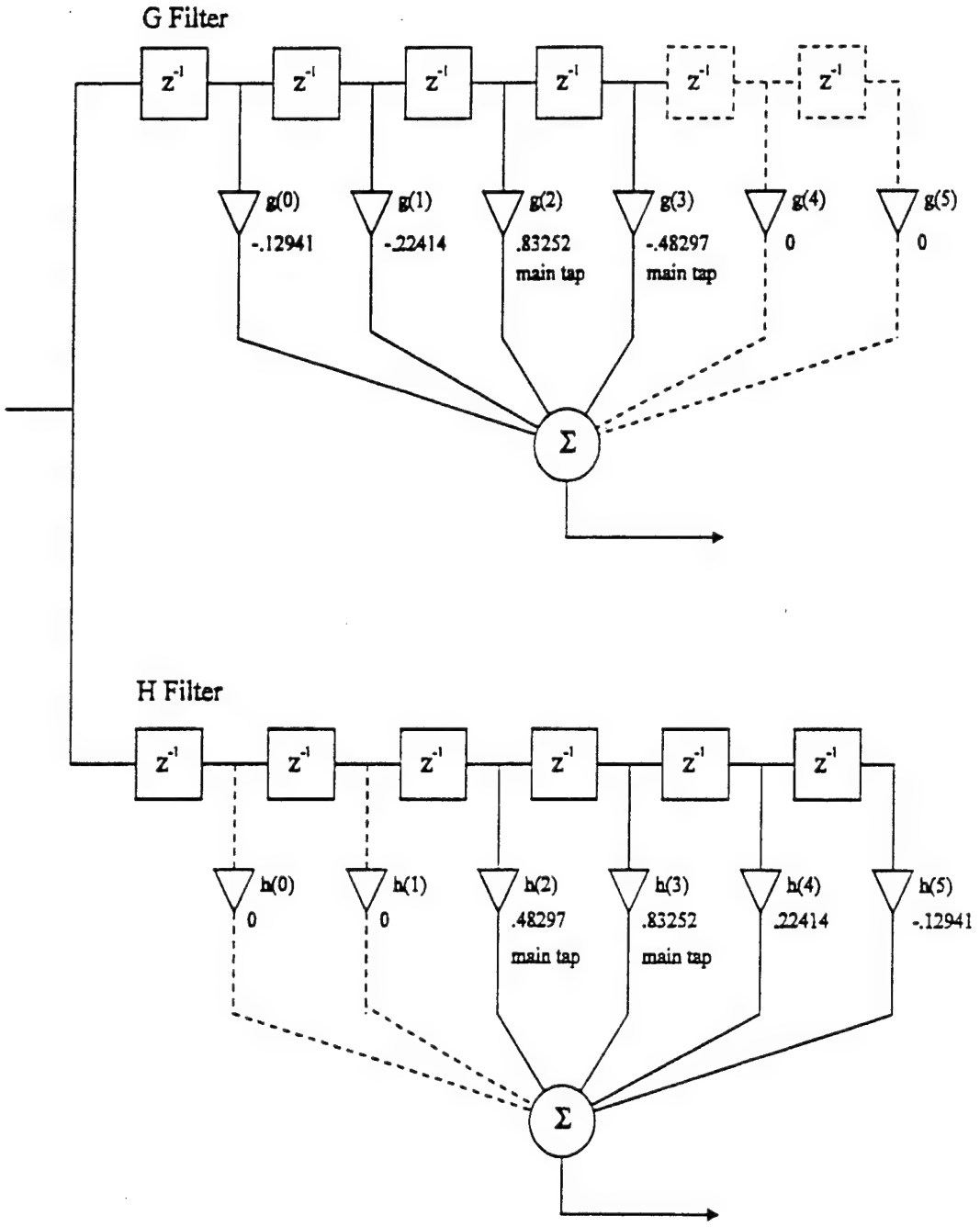



Figure 3.2. Finite Impulse Response Filters With Delays Added
(Tap Values Are For Daubechies' Four Coefficient Filter)

Another characteristic we need not consider is the regularity criteria mentioned in Chapter II. Daubechies' prototype filters meet this requirement by being "maximally flat" in the frequency domain around DC [12]. This does, indeed, eliminate the passband ripple. However, as

we will see, by ignoring regularity, we will have more freedom to design filters that better meet our other requirements.

**Analyzing Filters on the QMF Tree**

So far, we have only looked at the requirements for the individual H and G filters. We will also need to consider how these filters work together on a QMF filter tree like the one shown in Figure 2.16. To do this, it will be necessary to make a slight digression in this section. Here we will look at how we can take the string of H filters and decimators going down the low pass branch of the tree, and develop an equivalent filter.

One tool we will use is the "Z transform" [32] [36]

$$H(z) = \sum_n h(n)z^{-n} \tag{3.1}$$

where z is generally complex. Note that when we evaluate z along its unit circle, we can write $z = e^{j\omega}$ and (3.1) becomes

$$H(\omega) = \sum_n h(n)e^{-j\omega n} \tag{3.2}$$

the frequency response (Fourier Transform) of the filter. There are two properties of the z transform that we will need. First, just as with the DFT, a multiplication in the transform domain has the same effect as a digital convolution in the (discrete) time domain. Second, replacing z by $z^m$ in the transform domain is equivalent to inserting m zero valued samples between each sample of the sequence in the time domain [43].

This leads to a "noble identity," shown in Figure 3.3, that we will use to analyze the filter tree. What it says is that a filter, with a z transform $H(z)$, following a decimation by two, is mathematically equivalent to a filter with z transform $H(z^2)$, followed by a decimation by two. It is also true that a decimation by m followed immediately by a decimation by n is equivalent to a single decimation by mn [43].


Figure 3.3. Noble Identity

With this information, we can analyze the tree. Figure 3.4 shows this, beginning with the low pass branch of the tree (straightened for clarity). For the sake of example, we will consider a three layer tree. Taking the last filter and the preceding decimation, we apply the noble identity, leading to a string with two filters adjacent to each other followed by a decimation by four. We then convolve the filter coefficients as shown, remembering to insert zero valued coefficients for the second filter, and obtain an equivalent filter whose z transform we'll call $H'(z)$. We can then repeat this procedure (which is not shown in the figure), obtaining a filter $H'(z^2)$ adjacent to the first filter in the string, and another decimation by two adjacent to the decimation by four. Combining these as above, we end up with a single filter, which we will call $H_3$--the third layer equivalent low pass filter, and a decimation by eight.

In general, we will have a low pass filter $H_L$, where L is the last layer, followed by a decimation by $2^L$. The z transform of the equivalent filter is

$$H_L(z) = H(z)H(z^2)\cdots H(z^{2^{L-1}}) \tag{3.3}$$

In the time domain, each coefficient of the equivalent filter is

$$h_L(k) = \sum_m h_{L-1}(m)h(k-2m) \tag{3.4}$$

If N is the number of coefficients in the prototype filter

$$N_L = (2^L-1)N-(2^L-2) \tag{3.5}$$

will be the number of coefficients in the equivalent filter.

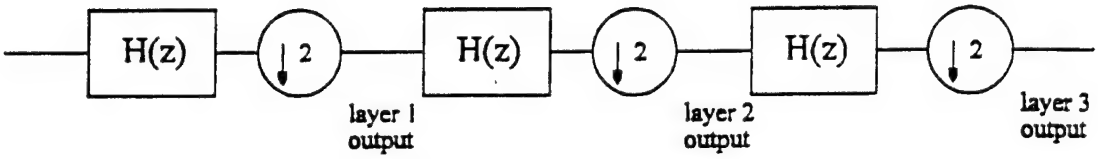
Taking the taps that were shown in Figure 3.2, we see we will obtain the equivalent second layer filter in Figure 3.5 (which happens to be Daubechies' four coefficient filter). Since the time frequency tiles from the second layer are four time samples long, we will have four main taps in the equivalent filter. In general, there will be $2^L$ main taps, and, for uniform detection across the length of the tile, we want these to be as equal in value to each other as possible. If we call mt the position of the first main tap in the prototype (starting with "0" for the position of the first tap), the position of the first main tap in the equivalent filter will be

$$mt_L = mt(2^L-1) \tag{3.6}$$

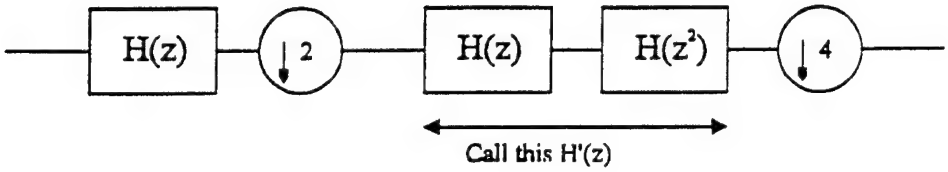
For many Wavelet filters, the main taps in the equivalent filter will not necessarily have the largest values (they don't, for example, in Figure 3.5 since $h_2(10) > h_2(6)$). However, the "best" filters for our purposes will. Notice relation (3.6) applies whether we begin counting taps at the first non-zero tap, or include the pure delay, as in the figures.

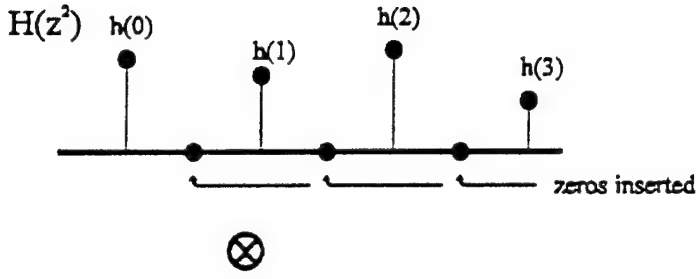
Of course, other branches of the QMF filter tree can be evaluated in a manner similar to that described above for the low pass filter. Instead of strictly using the z-transform of the H filter, the transform of the G filter would be substituted in the appropriate positions in (3.3) (depending on the particular path down the tree of the filter under evaluation). The recursive time domain equation (3.4) would also have to be modified. (3.5) and (3.6) remain unchanged.

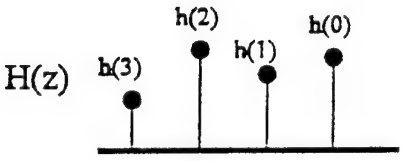
Low Pass Branch of QMF Bank Tree:




Equivalent:




=

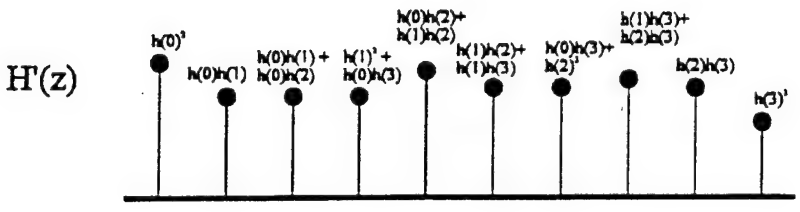



Figure 3.4a. Three Layer Low Pass Filter, and How to Analyze it

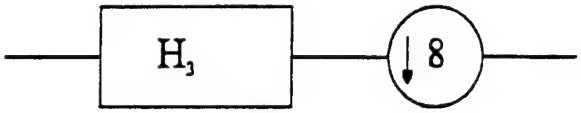



Figure 3.4b. Result of Analyzing Three Layer Low Pass Filter

**Common Wavelet Filters**

In this section, we will look at several Wavelet filters commonly discussed in the literature and see how they measure up to our requirements as specified above.

**The Haar Filter**

This is the filter discussed briefly in Chapter II. It has two coefficients

(3.7) $$h(0) = h(1) = 1/\sqrt{2}$$

It is the only Wavelet FIR filter that is symmetric. Since it has only two coefficients, both are main taps, and the filter perfectly meets our criteria for concentrating energy in time. It is easy to see the equivalent filters for any layer of a tree constructed with Haar filters will also consist entirely of main taps, and will perfectly concentrate energy in time.

In frequency, however, the Haar filter is not good. Figure 3.6 shows the frequency response for both the H and G filters. Obviously, large amounts of high frequency energy may be passed by the H filter, and vice-versa for the G filter. The situation is worse when the frequency responses of the outputs of the higher layers are examined, as in Figure 3.7. (More accurately, these are the outputs of the equivalent filters, and do not include the decimation by eight. For the low pass filter, the response is found by substituting $z = e^{j\omega}$ into (3.3). For other filters, (3.3) is first modified by replacing H(z) by G(z) in appropriate places, depending on the specific branch of the tree, and substituting $z = e^{j\omega}$.)

**Daubechies Four Coefficient Filter**

The four coefficient filter Daubechies presents in [12] is

(3.8)
$$\begin{aligned} h(0) &= .482962913145 \quad \text{main tap} \\ h(1) &= .836516303738 \quad \text{main tap} \\ h(2) &= .224143868042 \\ h(3) &= -.129409522551 \end{aligned}$$

where we declare the largest two taps to be the main taps. This is the filter shown in Figure 3.2, with two zero value taps added to the beginning of the filter to move the main taps to the center.

The frequency response at layer one is shown in Figure 3.8, and at layer three in Figure 3.9. It is easy to see that, although it is better than the Haar filter, much improvement is still desirable.

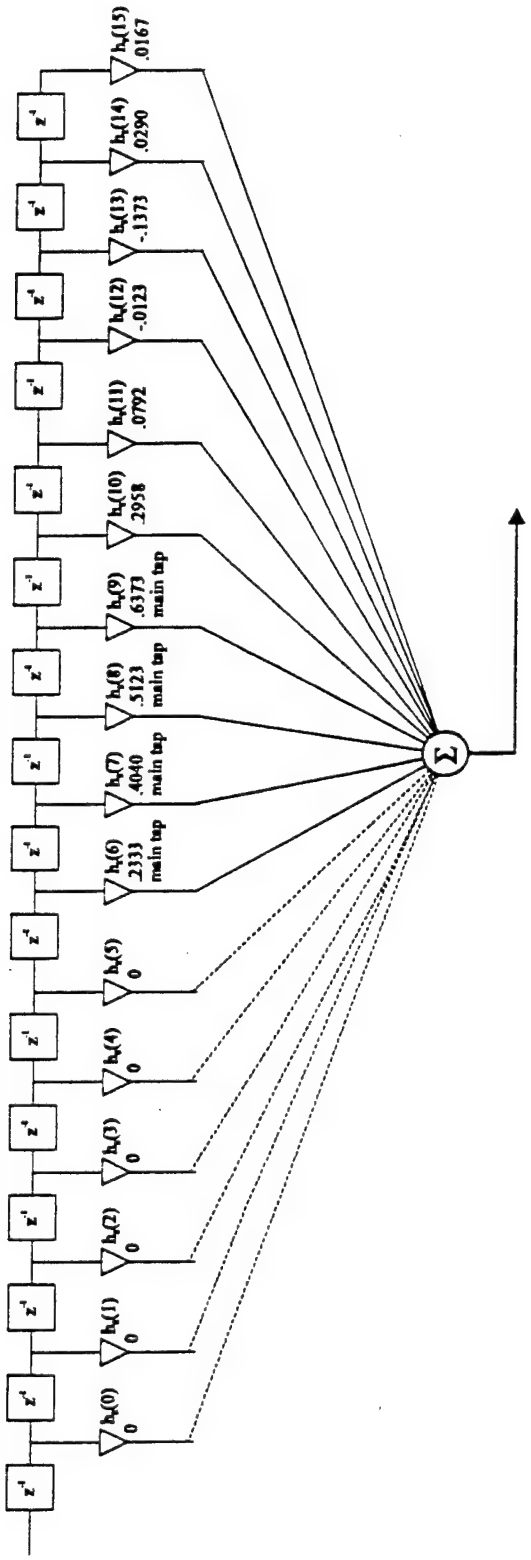



Figure 3.5. Second Layer Equivalent FIR Filter

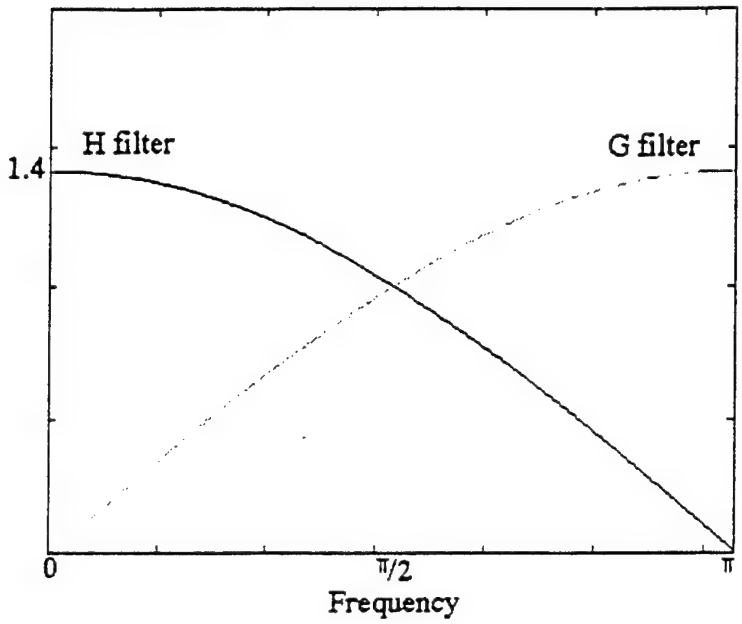



Figure 3.6. Haar Filter Magnitude Response

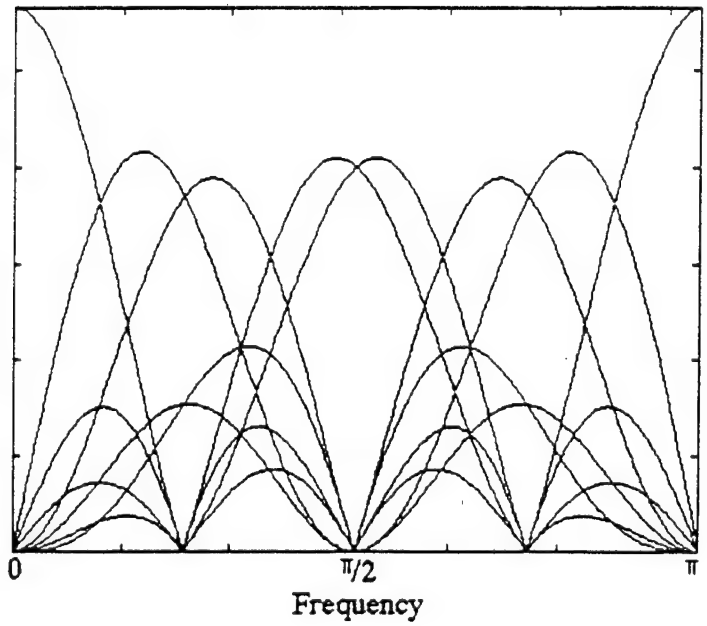



Figure 3.7. Layer Three Magnitude Response of Haar Filter Tree

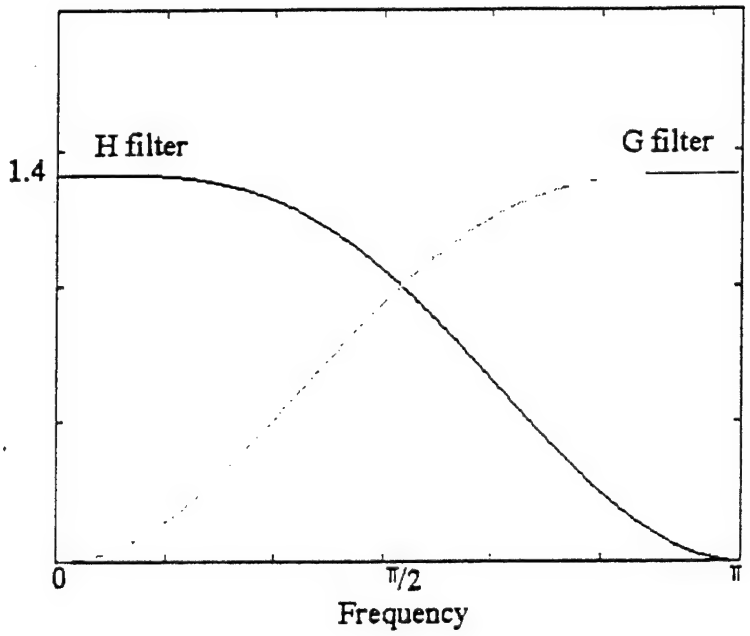



Figure 3.8. Daubechies' Four Coefficient Filter Magnitude Response

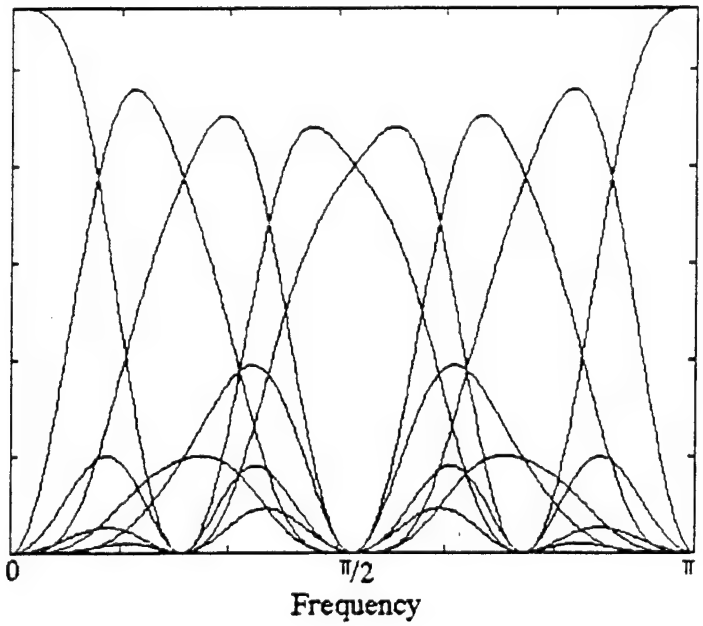



Figure 3.9. Layer Three Magnitude Response of Daubechies' Four Coefficient Filter Tree

**Daubechies 16 Coefficient Filter**

The 16 coefficient filter Daubechies presents in [12] is

$$
\begin{aligned}
h(0) &= .054415842243 \\
h(1) &= .312871590914 \\
h(2) &= .675630736297 \quad \text{main tap} \\
h(3) &= .585354683654 \quad \text{main tap} \\
h(4) &= -.015829105256 \\
h(5) &= -.284015542962 \\
h(6) &= .000472484574 \\
h(7) &= .128747426620 \\
h(8) &= -.017369301002 \\
h(9) &= -.044088253931 \\
h(10) &= .013981027917 \\
h(11) &= .008746094047 \\
h(12) &= -.004870352993 \\
h(13) &= -.000391740373 \\
h(14) &= .000675449406 \\
h(15) &= -.000117476784
\end{aligned}
\tag{3.8}
$$

where, again. we declare the largest two taps to be the main taps. In this case, ten zero valued coefficients will have to be added to the beginning of the H filter to center the main taps.

The frequency response at layer one is shown in Figure 3.10, and at layer three in Figure 3.11. Here, we see, the situation improves, although there is still a large transition region.

**The Sinc Filter**

The Haar filter perfectly concentrates energy in time. Is there an analogous filter that does the same in frequency? Yes, but it has an infinite number of coefficients. Here we will look at a way of modifying it slightly to make it practical.

If we start with an idea of the frequency response we want: flat passband, infinitely narrow transition, and zero across the stop band, and we take the inverse Fourier Transform of this, we will have a sinc function (2.7) in the time domain. Since our passband ranges from $-\pi/2 < \omega < \pi/2$ or $-0.25 < f < 0.25$, the nulls of the sinc function will be at 2T for a sampling rate of T.

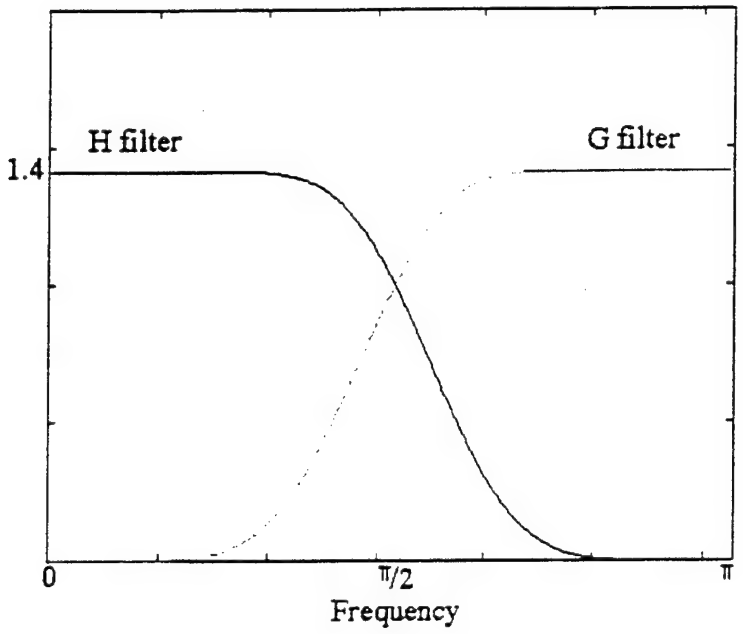



Figure 3.10. Daubechies' 16 Coefficient Filter Magnitude Response

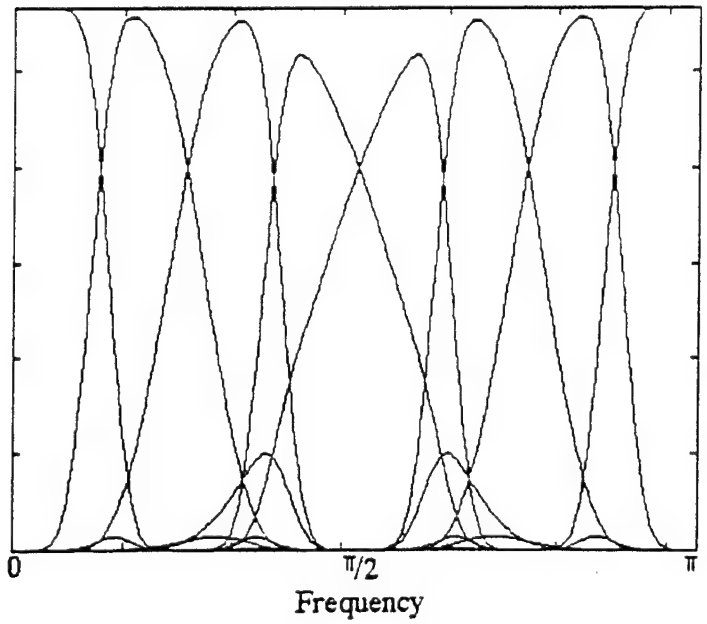



Figure 3.11. Layer Three Magnitude Response of Daubechies' 16 Coefficient Filter Tree

To obtain our filter coefficients, we want to sample the sinc function at the normalized sampling rate of T = 1, giving us a situation like that in Figure 3.12. One way to sample the function would be to let the main tap sample occur at the center of the main lobe. However, as we discussed informally above (and will see formally later), we must have two main taps, and desire their sum to be as large as possible. For the sinc function, this occurs if both of our main tap samples are equally spaced about the center of the main lobe. We also want the sum of the squares of the coefficients to be unity. This is achieved by scaling the sinc by $1/\sqrt{2}$. This is what is described by

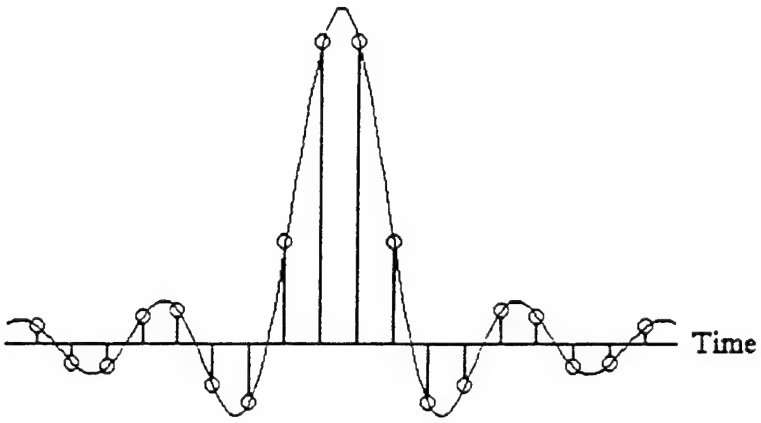



Figure 3.12. Sampling Under a Sinc Envelope

$$h(n) = \frac{1}{\sqrt{2}}\,\mathrm{sinc}\left(\frac{n+0.5}{2}\right) \qquad n \in \text{integers} \tag{3.9}$$

As it happens, this filter meets all of the rules for Wavelet filters. The only problem is that it has an infinite number of coefficients. (This, by the way, is how it is able to meet the Wavelet rules for orthogonality and still be symmetrical.) We now explore how to truncate this filter and maintain a good frequency response. The price we pay will be a small amount of non-orthogonality: there will be some cross-correlation between filters.

If we simply truncate the ends of the filter, we will have a frequency response like that shown in Figure 3.13, where 256 coefficients were used. The ripple in the passband, sometimes called the "Gibb's Phenomena" is a well known result of this type of truncation [33]. Since we are, in effect, multiplying the coefficients by a rectangular shaped window in the time domain, we can view this as a convolution of the perfect filter response with a sinc function (the Fourier Transform of the rectangular window) in the frequency domain.

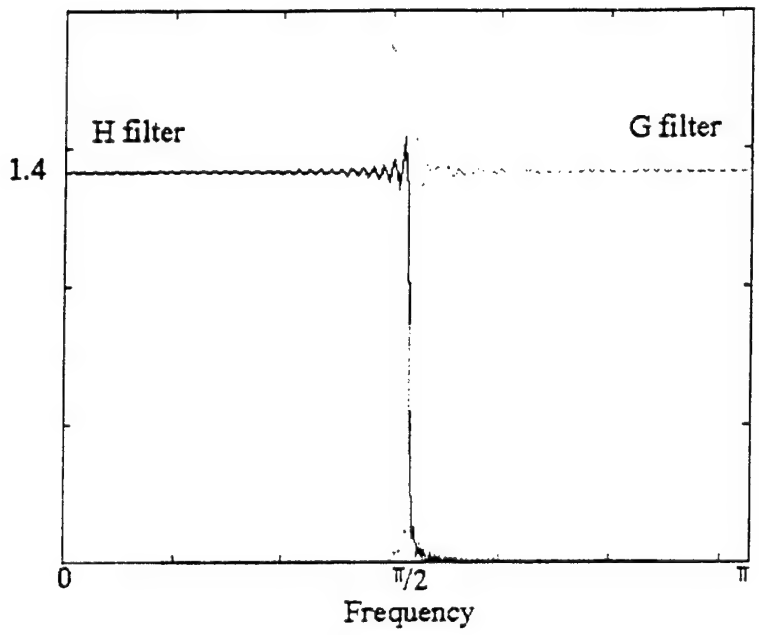



Figure 3.13. Magnitude Response of Truncated Sinc Filter

The solution to this is to use a non-rectangular window: one whose Fourier Transform has a narrower main lobe and smaller sidelobes than the sinc function. One that is commonly used is the Hamming window [33]. Multiplying the coefficients from (3.9) by this window, and using the results in a FIR filter, gives us the frequency response in Figure 3.14.

Recall that one of the results of orthonormality is the PC property: The H and G filters must meet (2.95), which we repeat here

$$|H(\omega)|^2 + |G(\omega)|^2 = 2 \tag{3.10}$$

Figure 3.15 shows the left hand side of (3.10) for the windowed and truncated filters, and it is clear that energy is lost at the filter transitions. This is primarily the result of the loss of orthogonality from truncating the filter. For detection, we would rather not lose the energy at those frequencies; a better trade-off would be to have a small amount of cross-correlation between the filters, so some energy shows up in more than one tile.

Intuitively, we should be able to achieve this by causing the prototype filter to have a passband that is slightly greater than $\pi/2$, so the H and G filters squeeze together slightly. This can be achieved by compressing the sinc envelope of (3.9) slightly. At the same time, it will be desirable to rescale the coefficients slightly, so the sum of the squares equals one. These modifications to (3.9) give us

$$h(n) = \sqrt{\frac{S}{2}}\,\mathrm{sinc}\left(\frac{n+0.5}{C}\right)w(n) \tag{3.11}$$

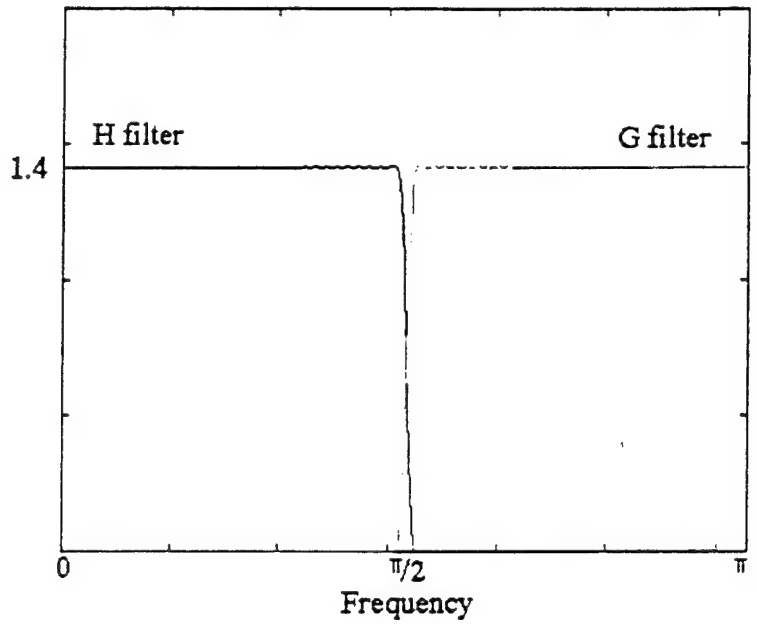



Figure 3.14. Magnitude Response of Hamming Windowed Truncated Sinc Filter

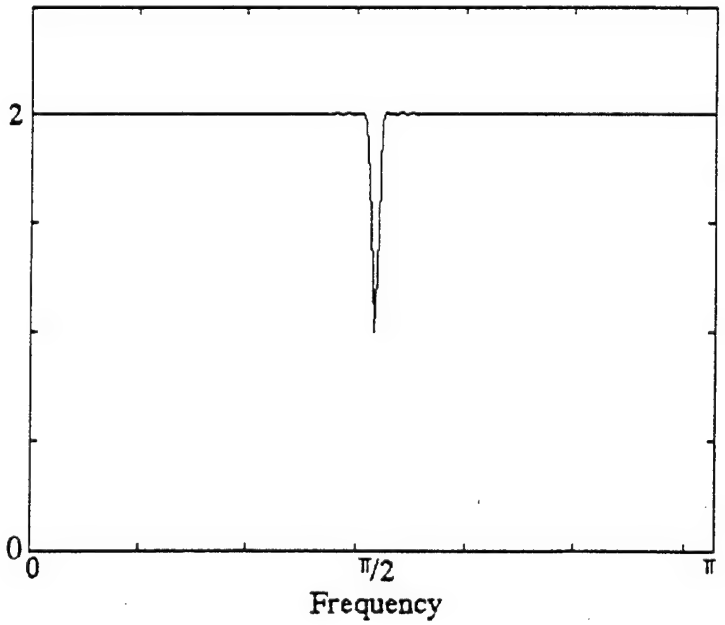



Figure 3.15. $|H(\omega)|^2 + |G(\omega)|^2$ For the Windowed Truncated Sinc Filter

where C is the compression variable, S is the scaling variable, and w(n) is the Hamming window. S and C were solved iteratively in the following manner: First C was adjusted to make $|H(\pi/2)| = 1$. Then the coefficients were computed and S was set equal to

$$\frac{1}{\sum_n [h(n)]^2}$$

The process was then repeated until C converged. Values of C and S, for various numbers of coefficients, N, are shown in Table I. As expected, as the number of coefficients increases, C approaches two, and S approaches one. The PC results are shown in Figure 3.16, and we see the energy at the transitions is passed, although there is a little ripple.

**Table II**

Values of C and S, For Various Numbers of Coefficients, N,
For the Modified Sinc Filter

| N | C | S |
|---|---|---|
| 16 | 1.80745449451012 | 1.22099338489190 |
| 32 | 1.90189442447347 | 1.10429475163860 |
| 64 | 1.95048099151118 | 1.05069751126202 |
| 128 | 1.97512319592231 | 1.02499785161815 |
| 256 | 1.98753217974055 | 1.01241259545240 |
| 512 | 1.99375872328059 | 1.00618488680080 |
| 1024 | 1.99687751845516 | 1.00308711196069 |
| 2048 | 1.99843829823671 | 1.00154222577172 |
| 4096 | 1.99921903384616 | 1.00077078066449 |
| 8192 | 1.99960948810196 | 1.00038530731821 |
| 16384 | 1.99980473684531 | 1.00019263291076 |
| 32768 | 1.99990236662119 | 1.00009631126893 |

For these filters, the greatest cross correlation occurs between tiles in the same frequency band and adjacent in time. This was measured, and for $N \geq 512$ it was found that

(3.12) $$\sum_n h(n)h(n+2) < 0.001$$

and decreases as N is increased. For our purposes, we can regard these filters as orthonormal.

For comparison with the Haar and Daubechies filters, Figure 3.17 shows the frequency magnitude response for the third layer of a modified sinc filter (with N = 512).

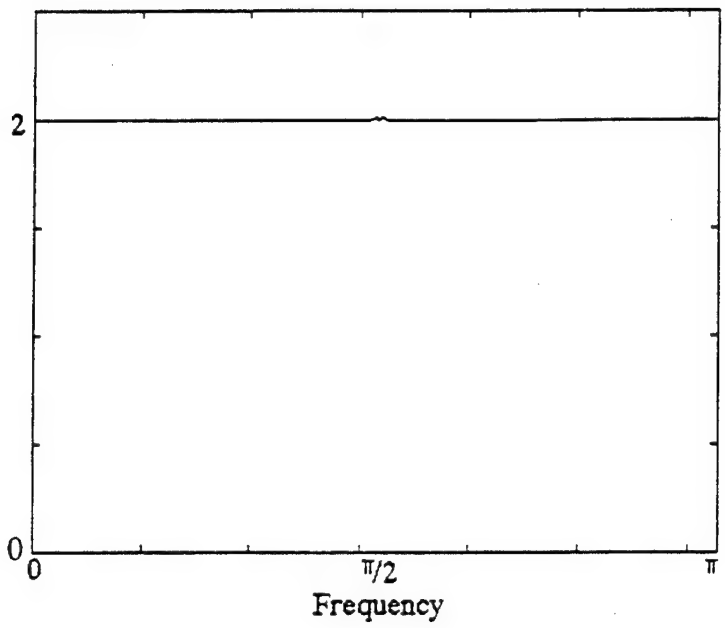



Figure 3.16. $|H(\omega)|^2 + |G(\omega)|^2$ For the Modified Sinc Filter

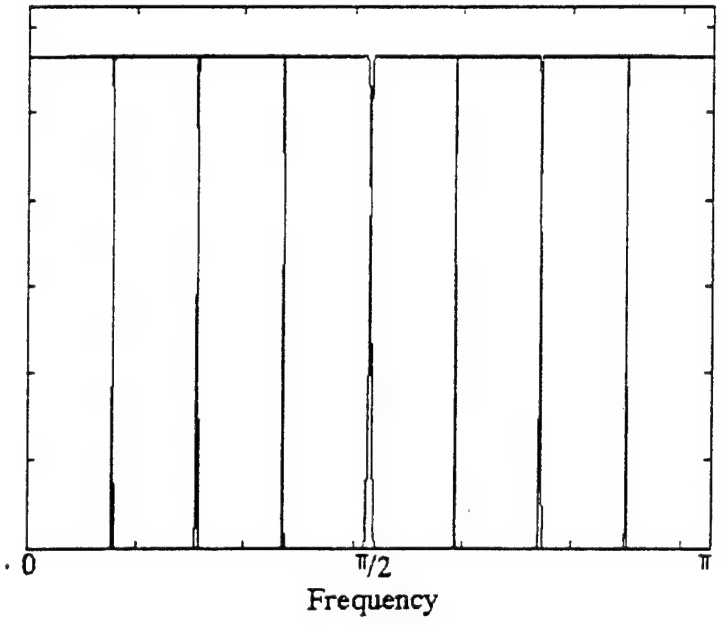



Figure 3.17. Layer Three Magnitude Response of
Modified Sinc Filter Tree

The trade-off for the good frequency response is a poor time domain response. Looking back at Figure 3.12, we see the taps adjacent to the main taps are also fairly large. In the next section, we will develop a method for formally looking at the out of tile energy passed by filters.

## Wavelet Filters For Energy Detection

Having looked at our filter requirements somewhat informally, we found the Haar filter has good time resolution and the sinc filter has good frequency resolution. Now we ask the question: can we develop Wavelet filters that are a compromise? Ones that have the best possible combination of time and frequency characteristics? The answer, of course, is "yes." Our strategy will be to first develop equations for the amount of energy a filter collects outside the time limits of the tile, and the amount of energy it collects outside the frequency limits. These will then be combined to form an objective function. Finding the best filters will then be a matter of minimizing the objective function subject to the Wavelet filter rules listed in Chapter II. As an additional benefit, the objective function can be used as a numerical comparison of other Wavelet filters, to judge how well they will work for energy detection.

### Out of Time Energy

Consider the L layer equivalent filter in Figure 3.18. The energy of the sequence output from the decimator is

$$\sum_{n'} [z(n')]^2 \tag{3.13}$$

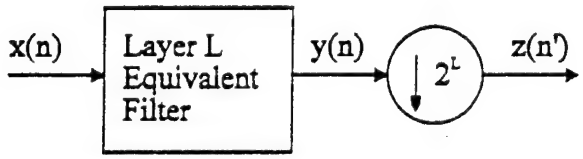



Figure 3.18. Layer L Equivalent Filter and Decimator

where n' are the sample times, n, that are passed by the decimator. Now, let's define an input sequence with a total energy of one, and with the energy uniformly distributed over the length, in time, of the tile of interest. This gives us

$$x(n) = \begin{cases} \dfrac{1}{\sqrt{2^L}} & n_0 \le n \le n_0 + 2^L - 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

where $n_0$ is such that the non-zero elements of x(n) are multiplied by the equivalent filter's main taps at a time sample passed by the decimator. We can call the passed sample time $n'_0$ and the energy of z

at that time will be the amount of energy within the tile's time dimensions passed by the filter. This gives us the following expression for "in time energy" for the layer L equivalent low pass filter

$$\alpha_L^2 = [z(n_0')]^2 = \left[\sum_{n=m_L}^{m_L+2^L-1} h_L(n)\left(\frac{1}{\sqrt{2^L}}\right)\right]^2 = \frac{1}{2^L}\left[\sum_{n=m_L}^{m_L+2^L-1} h_L(n)\right]^2 \tag{3.15}$$

We will find it more convenient to work with out of time energy, however. Since the total energy in x is one, and since the filters are orthonormal, the out of time energy is

$$E_L = 1-\alpha_L^2 \tag{3.16}$$

Looking at (3.15) and (3.16), we see the amount of out of time energy is minimized by increasing the sum of the main tap coefficients. It is also apparent that amount of out of time energy depends on the layer examined.

**Out of Frequency Energy**

To evaluate the out of frequency energy passed by the equivalent low pass filter, we need to look at the output preceding the decimator. (Decimation has the effect, in the frequency domain, of rescaling the magnitude response so the value previously at $\pi/2$, for example, will be moved to $\pi$ [43]. Any energy passed by the H filter above this frequency is aliased, as is any energy below this frequency passed by the G filter. This is the specific mechanism distributing out of frequency energy among "incorrect" filters in the tree.) In our analysis here, we will only consider the low pass branch of the tree.

Consider an input to the equivalent filter consisting of an impulse

$$x(n) = \begin{cases} 1 & n = n_0 \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

where $n_0$ is just a specific time. Then, the impulse response of the filter is its Fourier Transform

$$H_L(\omega) = \sum_n h_L(n)e^{-j\omega n} \tag{3.18}$$

Using (3.18), the amount of energy passed from the impulse that is within the tile's frequency band will be

$$\beta_L^2 = \frac{\int_{-\frac{\pi}{2^L}}^{\frac{\pi}{2^L}} |H_L(\omega)|^2 d\omega}{\int_{-\pi}^{\pi} |H_L(\omega)|^2 d\omega} \tag{3.19}$$

which can be simplified as follows. First, from (3.18), we can derive

(3.20) $$|H_L(\omega)|^2 = \left[\sum_k h_L(k)e^{-j\omega k}\right]\left[\sum_i h_L(i)e^{-j\omega i}\right] = \sum_k\sum_i h_L(k)h_L(i)e^{-j\omega(k-i)}$$

Looking at the denominator of (3.19), we find

(3.21)
$$\int_{-\pi}^{\pi}|H(\omega)|^2 d\omega = \sum_k\sum_i h_L(k)h_L(i)\int_{-\pi}^{\pi}e^{-j\omega(k-i)}d\omega =$$
$$\sum_k\sum_i h_L(k)h_L(i)\frac{2}{(k-i)}\sin[\pi(k-i)] =$$
$$2\pi\sum_k\sum_i h_L(k)h_L(i)\mathrm{sinc}(k-i) =$$
$$2\pi\sum_k h_L(k)^2 = 2\pi$$

where the last line is the result of applying (3.4) in

$$\sum_k h_L(k)^2 = \sum_k\left[\sum_m h_{L-1}(m)h(k-2m)\right]\left[\sum_n h_{L-1}(n)h(k-2n)\right] =$$
$$\sum_k\left[\sum_m\sum_n h_{L-1}(m)h_{L-1}(n)h(k-2m)h(k-2n)\right] =$$
$$\sum_m\sum_n h_{L-1}(m)h_{L-1}(n)\underbrace{\sum_k h(k-2m)h(k-2n)}_{\substack{0 \text{ when } m\neq n\\ 1 \text{ when } m=n}} =$$
$$\sum_m h_{L-1}(m)^2$$

(3.22)

which can be extended back to show

(3.23) $$\sum_k h_L(k)^2 = \sum_k h_{L-1}(k)^2 = \cdots = \sum_k h(k)^2 = 1$$

(3.19), then becomes

$$\beta_L^2 = \frac{1}{2\pi}\int_{-\frac{1}{2^L}}^{\frac{1}{2^L}} |H(\omega)|^2 d\omega =$$

$$\frac{1}{2\pi}\sum_k\sum_i h_L(k)h_L(i)\int_{-\frac{1}{2^L}}^{\frac{1}{2^L}} e^{-j\omega(k-i)}d\omega =$$

$$\frac{1}{\pi}\sum_k\sum_i h_L(k)h_L(i)\left[\frac{1}{k-i}\sin\left(\frac{\pi}{2^L}(k-i)\right)\right] =$$

$$\frac{1}{2^L}\sum_k\sum_i h_L(k)h_L(i)\,\mathrm{sinc}\left(\frac{k-i}{2^L}\right)$$

(3.24)

and we define the out of frequency energy to be

(3.25) $$S_L = 1-\beta_L^2$$

Here we have developed an equation to measure the out of frequency energy for the equivalent low pass filter for a particular layer. For layer one, the out of frequency energy for the high pass filter would be identical, because the magnitude responses for the H and G filters are mirror images. For higher layers however, the out of frequency energies of some branches of the tree will not, in general, be the same as for the equivalent low pass filter. The intuitive reason for this is that the H and G filters' passbands will not have perfectly flat magnitude responses, and, therefore, different amounts of energy will be passed at different frequencies. The only perfect solution for this is to use the Sinc filters, but, as we discussed earlier, they have poor out of time energy characteristics.

It should also be possible to develop equations like (3.24) for each equivalent filter for a particular layer, and to incorporate them (perhaps as some sort of average) into an objective function. We will not do that here, both because the calculations would be numerically intensive, and because we will be interested in filters that have "good" characteristics over a number of layers and branches. As we will see, minimizing the objective function we are about to describe for layer one will give us good filters.

**Objective Function**

From (3.16) and (3.25) we can define an objective function

(3.26) $$U_L = E_L + S_L = 2-\left(\alpha_L^2+\beta_L^2\right)$$

Figure 3.19 shows what this function really means. Energy passed by the filter that is within the time span of the tile, but outside the frequency band will be included (in $E_L$), as will passed energy that is within the band but outside the time span (in $S_L$). Energy passed by the filter that is both out of the time span and frequency band will be covered twice. The objective function then places an upper bound on the amount of out of tile energy, $OTE_L$, for the low pass equivalent filter for a particular layer of the tree

(3.27) $$0 < OTE_L \leq U_L$$

where equivalence only occurs for the Haar and Sinc filters.

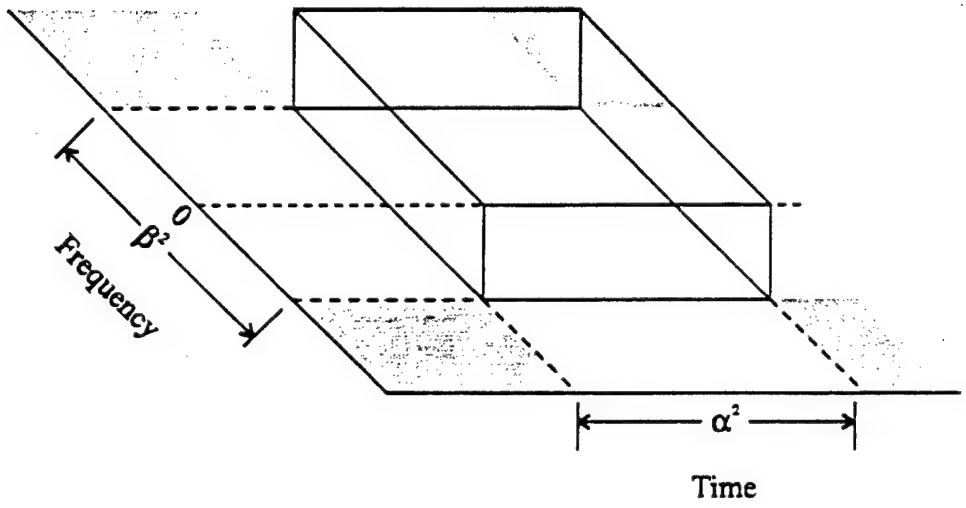



Figure 3.19. Tile in the Time Frequency Plane. Shaded Region Shows the Overlap of Energy in Objective Function

As mentioned above, one of the uses for (3.26) is as metric to decide how well a particular Wavelet filter will suit our purposes. Table II lists the values of $U_L$ for various layers, for the filters discussed previously.

The main reason for finding (3.26), however, is to try using it to find the best filters possible. To do this, we wish to minimize $U_L$ with respect to the prototype filter's coefficients, subject to the Wavelet constraints. One way to do this is to find the partial derivatives of (3.26) with respect to each of the prototype filter coefficients, applying the Wavelet constraints with Lagrange multipliers, setting the equations to zero, and solving simultaneously. Unfortunately, the equations turn out not to be linear. Iterative solution techniques were tried, but with only limited success: particularly for layers greater than two, the equations often failed to converge, even when various different initial guesses were tried.

**Table III**

Values of the Objective Function For Various Filters

| Layer | Haar | Da4[1] | Da16[1] | Sinc[2] |
|---|---|---|---|---|
| 1 | 0.1817 | 0.2581 | 0.2688 | 0.1874 |
| 2 | 0.2157 | 0.3389 | 0.4655 | 0.2252 |
| 3 | 0.2237 | 0.3771 | 0.5876 | 0.2343 |
| 4 | 0.2257 | 0.3963 | 0.6517 | 0.2366 |
| 5 | 0.2261 | 0.4060 | 0.6839 | 0.2373 |
| 6 | 0.2263 | 0.4108 | 0.6999 | 0.2376 |

Notes:

[1] Daubechies' four and 16 coefficient filters

[2] Hamming windowed, truncated, 512 Coefficient Sinc Filter

Fortunately, another method is viable. Unconstrained minimization problems are generally easier than multi-dimensional root finding [39]. A question then becomes whether it is possible to somehow parameterize the equations, replacing the constrained prototype coefficients with (fewer) unconstrained parameters. This very problem is solved in [47].

**Parameterizing the Wavelet Coefficients**

The relevant rules for finding Wavelet coefficients from Chapter II can be re-written as a series of equations which should be set equal to zero

$$\varphi_1 = \sum_n [h(n)]^2 - 1$$

$$\varphi_2 = \sum_n h(n)h(n+2)$$

$$\varphi_4 = \sum_n h(n)h(n+4)$$

$$\vdots$$

$$\varphi_i = \sum_n h(n)h[n+2(i-1)]$$

$$\vdots$$

$$\varphi_{\frac{N}{2}} = \sum_n h(n)h\left[n+2\left(\tfrac{N}{2}-1\right)\right]$$

$$\varphi_{\frac{N}{2}+1} = \sum_n h(n) - \sqrt{2}$$

$$\varphi_{\frac{N}{2}+2} = \sum_n g(n) = \sum_{n \text{ even}} h(n) - \sum_{n \text{ odd}} h(n)$$

(3.28)

where all but the last two come from the orthonormality requirement (Rule 2), and the last two come from the low pass/high pass filter requirements of Rule 5. (3.28) was used directly in the Lagrange multiplier problem described above. In [47], Zou and Tewfik use the same equations in their derivation of the following (which we write in the notation used in this paper)

$$\begin{bmatrix} H(z) \\ \left(z^{N-2}G\right)(z) \end{bmatrix} = \mathbf{V}(z)\mathbf{h}_0(z) \tag{3.29}$$

where

$$\mathbf{h}_0(z) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1+z \\ 1-z \end{bmatrix} \tag{3.30}$$

and

$$\mathbf{V}(z) = \left[\mathbf{I} + \left(z^2 - 1\right)\mathbf{v}_{N/2-1}\mathbf{v}_{N/2-1}^T\right]\cdots\left[\mathbf{I} + \left(z^2 - 1\right)\mathbf{v}_1\mathbf{v}_1^T\right] \tag{3.31}$$

where

$$\mathbf{v}_k = \begin{bmatrix} \cos\theta_k \\ \sin\theta_k \end{bmatrix} \tag{3.32}$$

(Actually, Zou and Tewfik also account for the Regularity Criterion. As stated earlier, however, we are not concerned with that, and so have not included that part of their work in (3.29)-(3.32) ).

With (3.29)-(3.32), the set of N coefficient Wavelet prototype filters map onto the set of N/2-1 values of $\theta \in [0, \pi)$. For our problem then, we simply have to minimize the objective function with respect to the $\theta$'s. The parameters we are allowed to select are the number of filter coefficients and the position of the first main tap.

**Numerical Results**

The objective functions in (3.26) for $U_1$ and $U_6$ were minimized for a number of filter lengths and main tap positions, using the parametric equations of (3.29)-(3.32) and the Neider-Meade simplex algorithm for minimization [39]. Specifically, the Matlab function FMINS.M was used [34]. For this method, initial guesses for the parametric values, $\{\theta\}$, were input to the Matlab function, and it eventually returned an answer.

The results were generally satisfactory, although some initial guesses of $\{\theta\}$ led to local minimums. This was usually obvious when it happened, since the main taps were not as large as they should be and the other taps were not all small. The possibility cannot be ruled out, however, that some local minimums exist, and were found, that lay close to the global minimum.

For mt = 0, good results were always found. Part of this success is due to the fact that when all of the $\theta$'s in (3.32) are set to $\pi/2$, the resulting filter coefficients are the Haar set with

$h(0) = h(1) = 1/\sqrt{2}$ and all other coefficients equal to zero. Since the coefficients we desire will be close to these values, FMINS.M was always able to find a good minimum. Other values of mt, however, often resulted in lower minimums. Unfortunately, in these cases, particularly for filters with larger numbers of coefficients, it was not always easy to make good initial guesses for $\{\theta\}$.

As it happens, for a given number of coefficients and value for mt, the coefficients to minimize $U_1$ and $U_6$ are not very different. Because of this, and the added computational complexity involved in computing the equivalent filter coefficients to find the objective function for higher layers, it was often a good idea to first minimize $U_1$, and then use the values found for $\theta$ as the initial guesses to minimize $U_6$.

As for specific results: $U_1$ was minimized for all even coefficients from four to 18, and for all possible values of mt (values of mt greater than half of the number of coefficients were not evaluated, since they simply result in coefficients that are time reversed). One thing to note here is that, for any given value of mt, a solution for N coefficients is a subset of the possible solutions for N+2 coefficients, since zero values for the last two coefficients is permissible. Therefore, for larger filters, the true global minimum should be equal to or less than the minimum for smaller filters.

Certain values of mt always seemed to work better than others. For greater than six coefficients in the filter, mt = 2 always worked best of all. For this reason, filters were found minimizing $U_1$ for mt = 2 for even coefficients greater than 18. This was done up to 24 where FMINS.M failed to find a solution. Figure 3.20 shows the values of $U_1$ found, versus the number of coefficients, and we see the curve appears to approach an asymptote, leading us to suspect $U_1 = 0.1166$, the value found for a 22 coefficient filter with mt = 2, is about the best we can do.

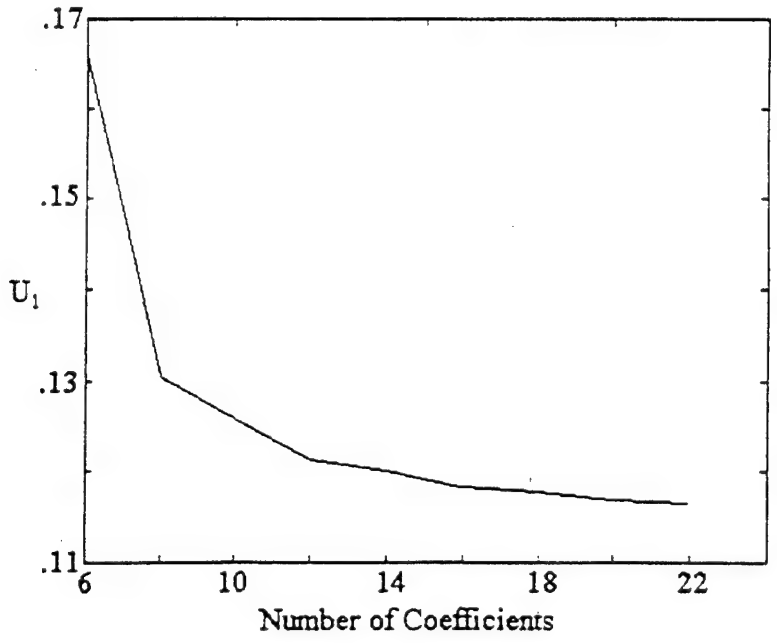



Figure 3.20. Minimum Values of $U_1$ Versus the Number of Coefficients, When mt = 2

Minimum values for $U_6$ were also found for different numbers of coefficients and values for mt, and the results were found in all cases to be similar to those for $U_1$. As before, the best configuration found was for a 22 coefficient filter with mt = 2. In this case, values for $U_1$ and $U_6$ are 0.1189 and 0.1554 respectively. Note this value for $U_1$ is only slightly higher than when it is minimized.

We conclude, then, the 22 coefficient filter with mt = 2, and with $U_1$ minimized, is a good one to use here. The coefficients are

$$
\begin{aligned}
h(0) &= -0.06937058780687 \\
h(1) &= \ 0.08787454185760 \\
h(2) &= \ 0.69303661557137 \quad \text{main tap} \\
h(3) &= \ 0.69307535109821 \quad \text{main tap} \\
h(4) &= \ 0.09491659263189 \\
h(5) &= -0.09139339673362 \\
h(6) &= -0.04245335524319 \\
h(7) &= \ 0.04675991002879 \\
h(8) &= \ 0.03613538459682 \\
h(9) &= -0.03468231058910 \\
h(10) &= -0.02281230449607 \\
h(11) &= \ 0.02100935003910 \\
h(12) &= \ 0.02296583051731 \\
h(13) &= -0.02145780458290 \\
h(14) &= -0.01348759528448 \\
h(15) &= \ 0.01318436272982 \\
h(16) &= \ 0.01550256127783 \\
h(17) &= -0.01636308107201 \\
h(18) &= -0.00627176286924 \\
h(19) &= \ 0.00993238693842 \\
h(20) &= -0.00105459770882 \\
h(21) &= -0.00083252852776
\end{aligned}
\tag{3.33}
$$

and the frequency responses for layers one and three of the QMF bank tree are shown in Figures 3.21 and 3.22.

Finally, Figures 3.23-3.25 compare the energy concentration abilities of the Haar, the modified Sinc (with 512 coefficients), and the filter in (3.33). Each figure shows the square of the output values of layer six of the QMF filter bank tree for a particular input signal.

The signal is a sinusoid with a frequency of 0.3, turned on for 128 seconds. This length of time would be covered by two tiles at layer six, however, the instants at which the signal was turned on and off was purposely set so as not to have any particular relation to the tiles.

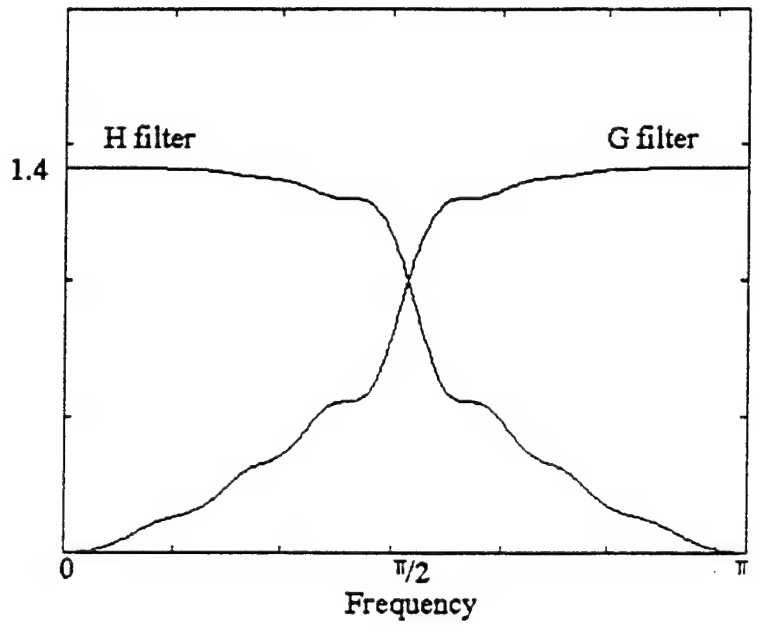



Figure 3.21. 22 Coefficient Energy Detection Filter, Magnitude Response

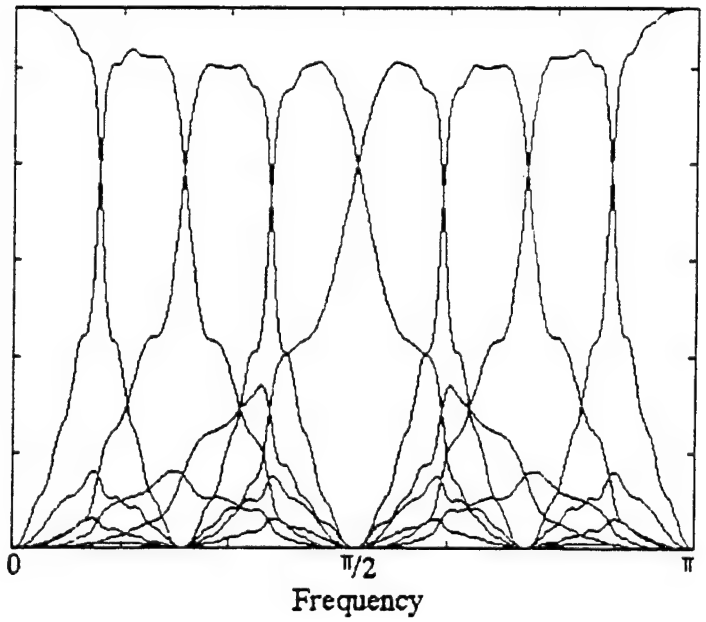



Figure 3.22. Layer Three Magnitude Response of 22 Coefficient Energy Detection Filter

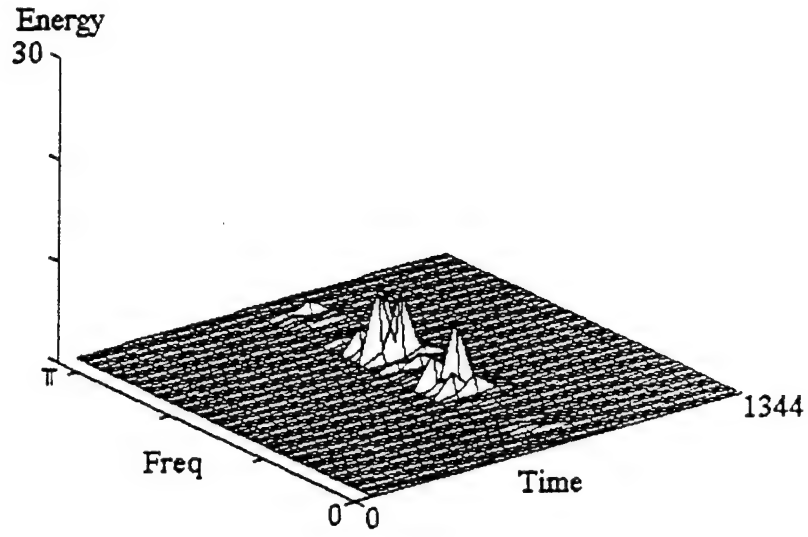



Figure 3.23. Cell Energy at Layer 6 With Haar Filter

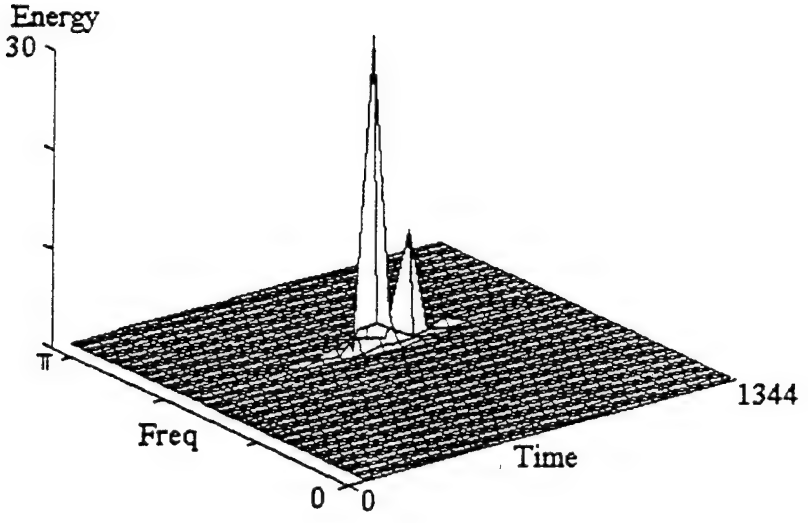



Figure 3.24. Cell Energy at Layer 6 With Sinc Filter

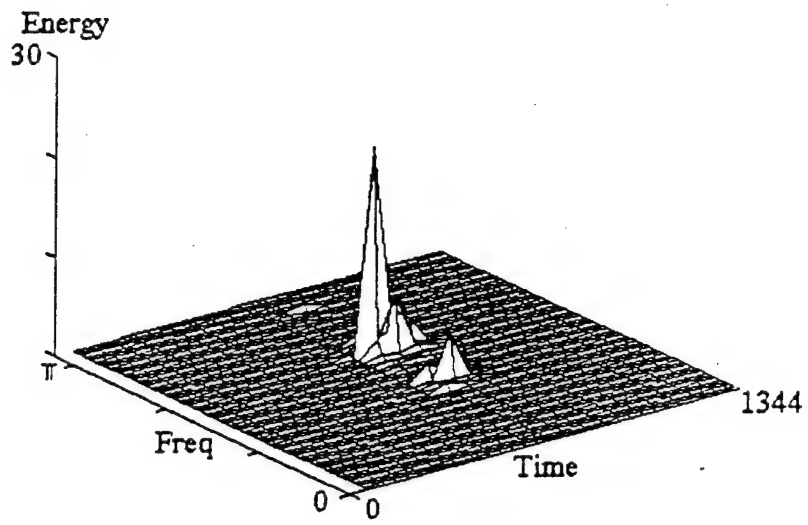



Figure 3.25. Cell Energy at Layer 6 With 22 Coefficient Energy Concentration Filter

As we can see, the Haar filter, as predicted, does a perfect job of isolating the signal "cell" in time, although it spreads the energy over the frequency domain. The Sinc filter, on the other hand, concentrates the signal well in frequency, but spreads it in time. The filter of (3.33) proves to be a good compromise.

**Summary and Conclusions**

In this chapter we examined informally, and then formally, our requirements for Wavelet filters with good tiling characteristics in the time frequency plane. We saw the Haar filter is the best we can do in the time dimension, while the Sinc filter (or rather, one of its realizable approximations) is the best in the frequency dimension. Each of these may be useful when one of either the time or frequency features of a signal are being examined.

# IV. Simulation Programs

## Introduction

In this chapter we look at some of the programs used in the Monte-Carlo simulations for this research. Simulations are used to verify mathematically derived results, and to provide results when the mathematics are intractable. All of these simulations were carried out on a 486DX2 based personal computer using Matlab for Windows version 4.0.

The organization of the simulation programs follow the receiver block diagram in Figure 1.2. A waveform is assumed to be properly band pass filtered and sampled at the Nyquist rate. The Matlab code for generating these sampled waveforms, consisting of spread spectrum signals and additive band limited white Gaussian noise (WGN), is described below. These signals are then input to a program, described later in this chapter, which carries out the function of the Quadrature Mirror Filter (QMF) bank tree. The output is then written to the hard drive.

The QMF output data, from the hard drive, can then be analyzed using code written to carry out the algorithms developed in this research. This code is discussed in later chapters, along with the specific simulations.

## LPI Signal Generators

The programs discussed in this section are listed in Appendix A, along with the script files used to produce Figures 4.1-4.6. Each program generates a sequence, with $2^{15} = 32768$ elements, representing a sampled signal. For convenience, we will generally assume a sampling rate normalized to one sample per second. Because of this, all of the frequency inputs for the programs in this section should be in the range of [0, 0.5) Hz.

Each of these programs generates and uses one or more sets of "random" numbers. Of course, as with any algorithmically generated set of numbers, these are not truly random, but are designed to pass a number of statistical tests for randomness [39]. Matlab uses the following to generate uniform "random" integers [34]

$$I_{j+1} = 7^5 I_j \bmod(2^{31} - 1) \tag{4.1}$$

where $I_j$ is the previous integer and $I_{j+1}$ is the new one. For a uniform distribution, the integers are divided by the maximum possible integer value to obtain numbers in the range of (0, 1.0). Gaussian distributions, like the one used in NOISEGEN.M, are then obtained from the linear distribution via a transformation algorithm built into Matlab. All of the programs discussed in this section are written so the initial seed value for each set of numbers is a user supplied input.

All of the programs (except NOISEGEN.M) generate a signal by modulating a sine function carrier. In order to combine signals, where the carrier phases generally will not be related to one another, **theta** is provided as a user input. It creates an initial phase offset in radians.

To construct an input sequence for the QMF bank, output sequences from the signal and noise generators are scaled to obtain correct signal to noise ratios, and added.

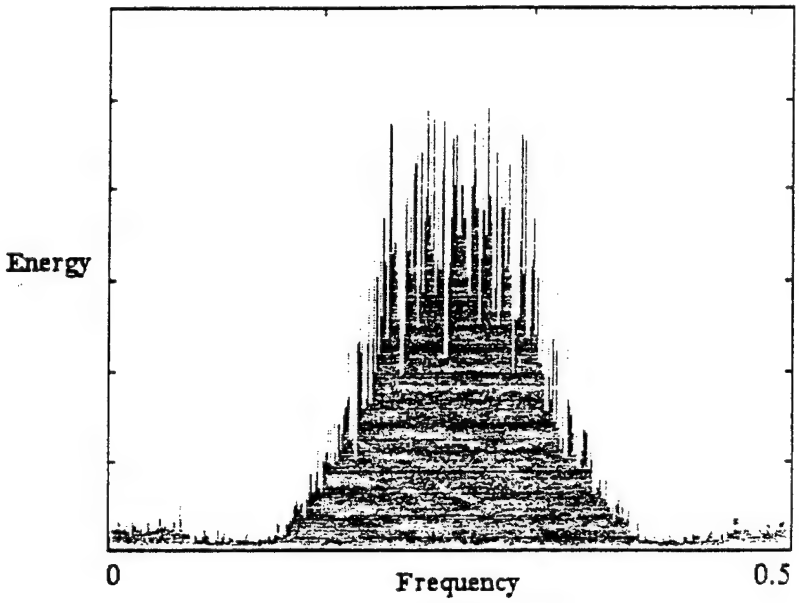



Figure 4.1. Fourier Transform of DS Signal

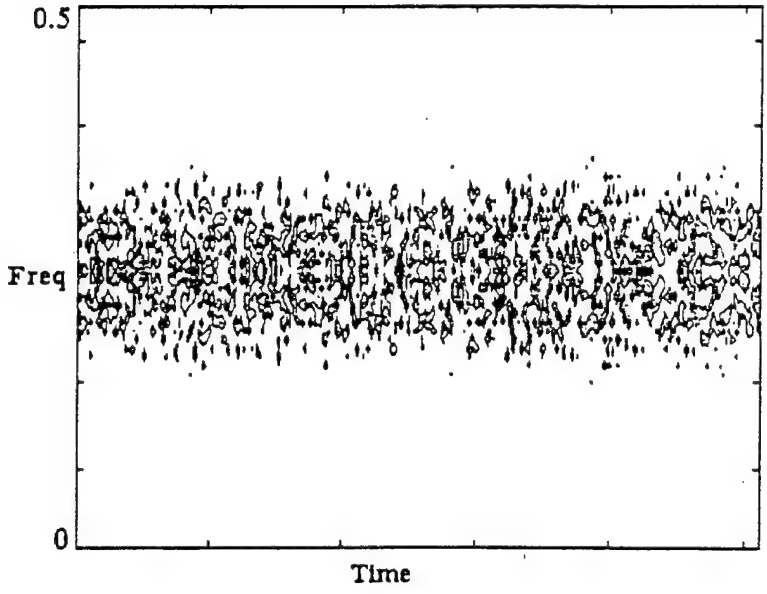



Figure 4.2. Short Time Fourier Transform (STFT) of DS Signal

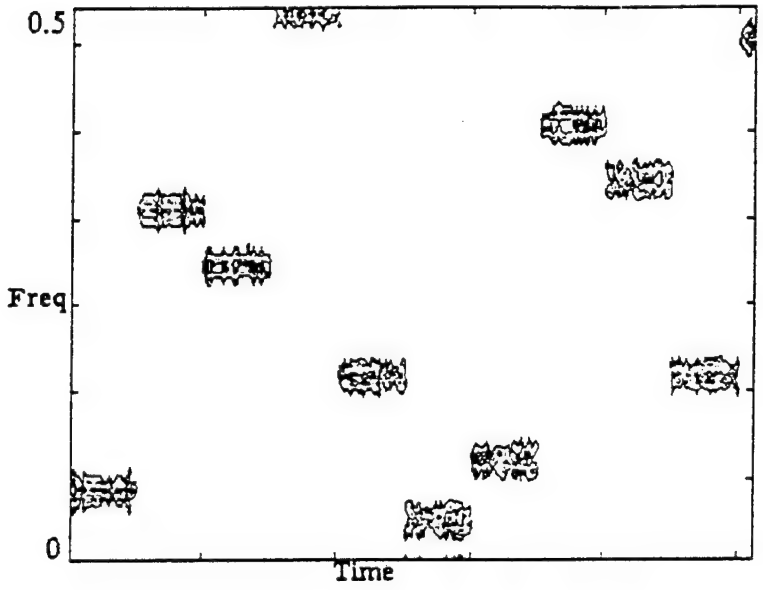



Figure 4.3. STFT of Fast FH/DS Signal

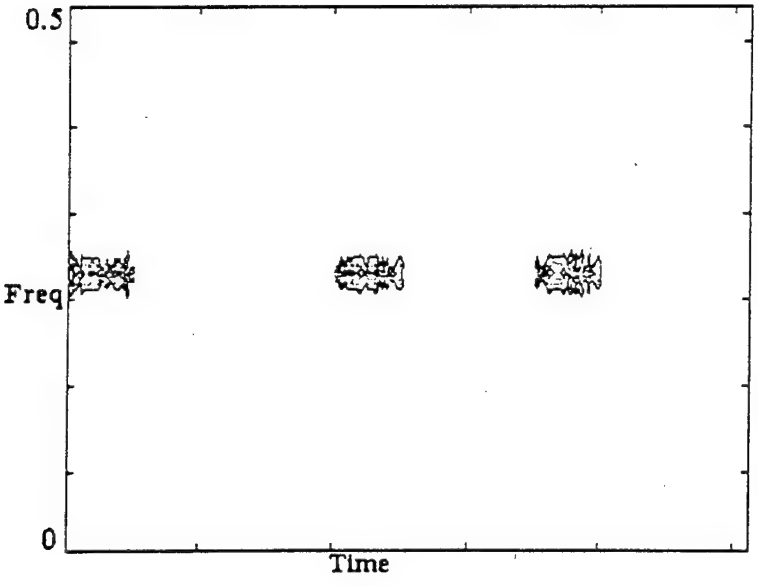



Figure 4.4. STFT of TH/DS Signal

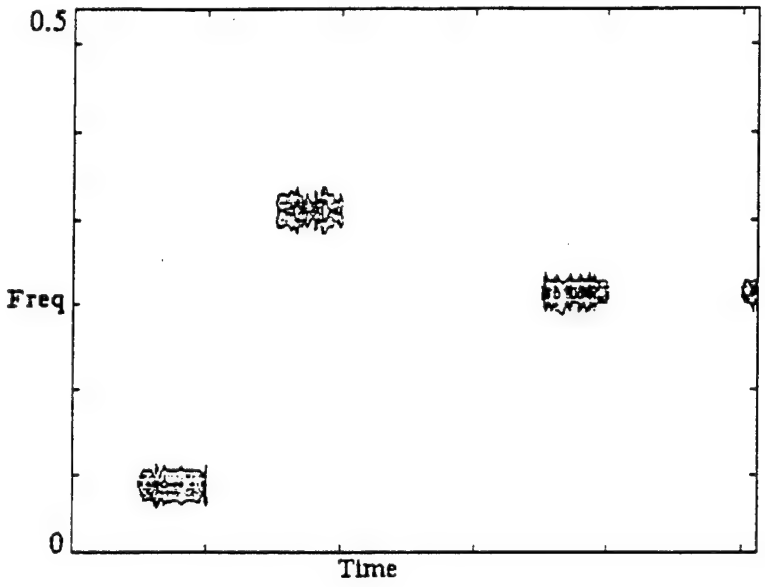



Figure 4.5. STFT of Fast FH/TH/DS Signal

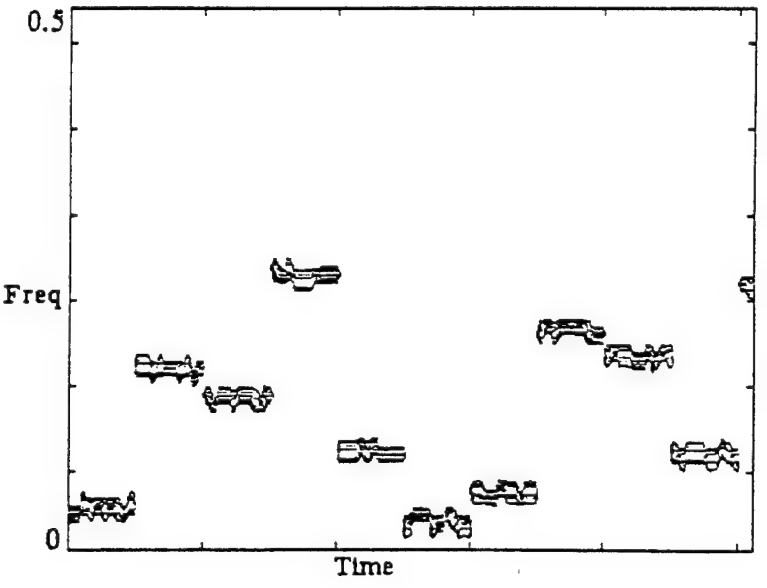



Figure 4.6 STFT of Slow FH/DS Signal

**Noise Generator**

NOISEGEN.M generates a sequence of numbers with a Gaussian distribution with mean of zero and variance of unity. The only input is a seed value for the random number generator.

**Direct Sequence (DS)**

DS.M generates a DS signal. The signal is spread using a random binary waveform. **fc** is the center frequency of the carrier, and **B** is the effective bandwidth of the spreading signal.

Figure 4.1 shows the Fourier Transform of the energy of a signal generated from DS.M with **fc** = 0.25 Hz and **B** = 0.15 Hz. The figure shows that the energy distribution is approximately a sinc-squared function as we would expect when the modulating signal is a random binary waveform. (The energy for this, and all other figures in this chapter, is plotted linearly, not in decibels.) The irregularity in the figure is due to the finite observation time. Essentially, we have the spectrum of a finite random binary waveform. It is important to note we are defining the signal's effective bandwidth such that the distance between the nulls at the ends of the main lobe (sometimes called the null to null bandwidth) is equal to two times **B**.

Figure 4.2 shows this same signal's energy using a Short Time Fourier Transform (STFT) with the signal broken up into non-overlapping 128 sample segments. The figure is plotted using Matlab's built in contour plot. Although it is hard to make quantitative measurements using this type of plot without knowing how each contour line is derived, the figure does show the general shape of the signal.

**Fast Frequency Hopped (FH) and Fast FH/DS**

FFH.M generates a Fast FH signal with, or without, DS spreading. DS spreading is included if the user supplies a value for **B**, the input variable representing bandwidth. If no value is supplied, the bandwidth is taken to be the inverse of the cell length. The cell length is determined by the input variable **T**.

The number of hop channels are determined by the input variable **CH**. The channels are adjacent to each other, each with bandwidth **B**. The lower edge of the first channel, equal to the center frequency of the first channel minus **B**/2, is determined from the input variable **lf**.

As with DS.M, the spreading sequence is a random binary waveform with a null to null bandwidth equal to two times the effective bandwidth, **B**.

If the input variable **phase** is set to zero, the first hop transition is at the very beginning of the signal sequence. When **phase** is non-zero, the signal is shifted to the left by **phase** samples.

Since the time bandwidth resolution of the STFT is equal to the time bandwidth product of a Fast FH cell, it is difficult to show a signal in this way. However, when DS spreading is added to the signal, the STFT decomposition gives a picture that is easy to interpret, and this is what is shown in Figure 4.3. As before, Matlab's built in contour plotting function is used. In this particular case, a 20 channel signal is shown, with cell length equal to 3200 seconds and channel bandwidth equal to 0.025 Hz, yielding a time bandwidth product of 80.

**Time Hopped (TH) and TH/DS**

TH.M generates TH and TH/DS signals, depending on whether a value is given for **B**. The inputs **T** and **phase** perform the same function as with FFH.M.

Each cell may be hopped to one of **slots** possible positions (analogous to channels in the frequency hopping format). The available slots are adjacent to each other in time.

**fc** is the center frequency for the signal. As with DS.M, the spreading sequence is a random binary waveform with a null to null bandwidth equal to two times **B**. If no value is given for **B**, the bandwidth will be taken to be equal to the inverse of the cell length.

Figure 4.4 shows a TH/DS signal, generated with TH.M, with center frequency of 0.25 Hz, cell length of 3200 seconds, bandwidth of 0.025 Hz, and with three possible hop slots per cell (not a particularly realistic number, but it gives us a picture with several cells).

**Fast FH/TH and Fast FH/TH/DS**

FFHTH.M generates Fast FH/TH and Fast FH/TH/DS signals. It is an amalgamation of the FFH.M and TH.M programs, and all of the input variables have the meanings described above.

With this signal format, a cell may be hopped to one of a number of positions in time and frequency. Figure 4.5 shows an example generated by FFHTH.M, where there are 20 channels in frequency and three slots in time. As before, the cell length is 3200 seconds and the bandwidth is 0.025 Hz.

**Slow FH**

SFH.M generates Slow FH signals. "Information" is generated randomly and modulates the signal using multiple frequency shift keying (MFSK). The number of frequencies are determined by the input variable **M**. The signal is then hopped among **CH** channels and each hop cell has a length of **T**, just as for a Fast FH signal. The input variables **lf** and **phase** also function identically to their counterparts in FFH.M.

Figure 4.7 is a conceptual example of a Slow FH cell showing concentrations of energy due to the FSK modulation, and indicating the meaning of some of the variables used in SFH.M. **M**, **T**, and **Ntpc** are input variables. **spt** is determined by dividing **T** by **Ntpc**. **bw** is taken to be the inverse of **spt**, and the distance between adjacent shift frequencies is taken to be equal to **bw**.

Figure 4.6 shows a Slow FH signal generated with SFH.M, with 20 channels, a cell length of 3200 seconds, four information channels (4FSK), and ten FSK transitions per hop. The figure was produced using a STFT with the signal broken into non-overlapping 128 sample segments. The resolution is not fine enough to determine the information modulation in the signal, but it is enough to see the energy concentration in the cells. In this example, the time bandwidth product of each cell is 40, less than the time bandwidth product of the signals shown in Figures 4.3-4.5.

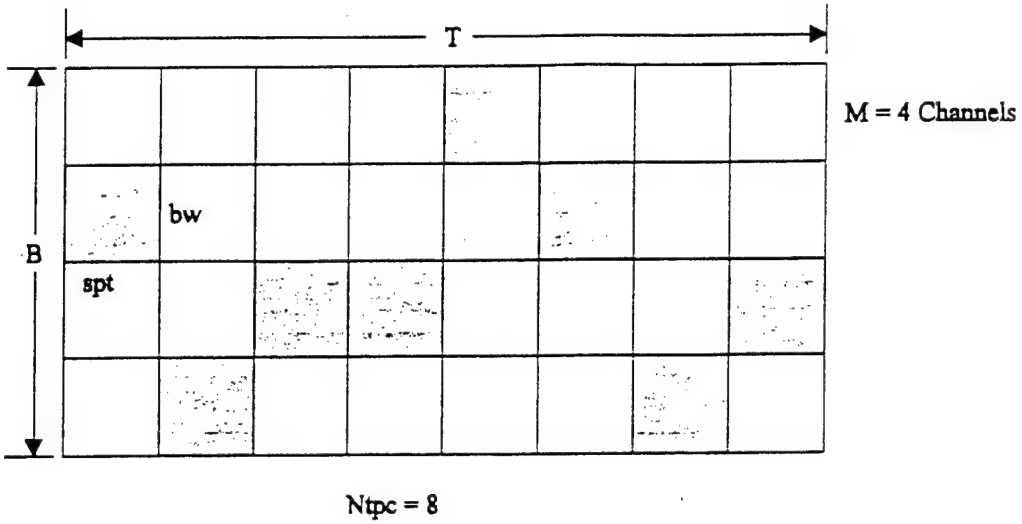



Figure 4.7. Conceptual Example of a Slow FH Cell

**Band Pass Filter**

The initial band pass filter in Figure 1.2 is not modeled in these simulations. To filter the signals generated by the programs described above would require a much greater sampling rate in order to simulate the analog signal on the computer. To avoid this, we note that although the spread spectrum signals are mathematically of infinite bandwidth, they are essentially bandlimited, as the energy in their sidelobes becomes insignificant for frequencies far from the center frequency. We will, therefore, restrict the frequency range of the input signal in our simulations to roughly the range 0.125 to 0.375 Hz, and will only examine outputs in that range. In this way, the signal energy outside the range [0, 0.5) Hz will be insignificant.

**Quadrature Mirror Filter Bank Tree**

QMF.M, in Appendix A, decomposes an input sequence as described in Chapter II for the QMF bank tree for arbitrary tiling. **f** is the input sequence, and **filter** is an optional string variable input used to specify the filter file. (If **filter** is not specified, the HAAR.M file with Haar filter coefficients are used.) Because the output for each succeeding layer takes an increasingly longer time to compute, and because the later layer outputs are not always needed, **N** is an optional input that dictates the last layer computed.

Output sequences from each layer of the bank are written to files C:\DATA\LAYERx.DAT on the hard drive (where x is replaced by the layer number). These are ASCII files, and the data is stored in double precision format as a matrix with each column representing the output from a particular filter in the layer. Frequency is represented across each line in the file (lowest frequency to the left), and time is represented down the file (lower representing later time).

Looking at the file listing in Appendix A: we see first that if an m file of filter coefficients is not specified, the Haar coefficients are used by default. The number of layers is then determined from the length of the input sequence or by N. The input matrix for each layer, **I**, is initially set equal to

the input sequence, and the output matrix for each layer, **out**, is set equal to **I** (in order to reserve space in memory).

The code then loops once for each layer of the QMF bank. Inside the loop, the layer number is first displayed on the screen to indicate to the user what the program is doing, and **flag** is initialized to 1. **out** is then reshaped to match the dimensions required for the output of the layer.

The code then loops again (nested inside the first loop) once for each column of the current input matrix. **G** and **H** are the output sequences from each filter pair. As described in Chapter II, these sequences represent a decomposition of the input into lower and upper frequency bands, and which is which alternates down the layer. **flag** keeps track of this, and the sequences are written to **out** in the correct order.

Finally, the output for the current layer is used as the input for the next, and the output data is written to disk.

HAAR.M, DA4.M, DA16.M, CON22.M, and TSINC.M are listed in Appendix A and are all m files that can be specified for use with QMF.M. HAAR.M uses the filter coefficients for the Haar filter, and DA4.M and DA16.M use the coefficients for the four element and 16 element Daubechies filters, respectively [12]. CON22.M uses the coefficients we found in (3.33) for a filter that minimizes the out of tile energy, and TSINC.M uses the coefficients of a modified sinc filter as described in Chapter III.

HAAR.M, DA4.M, DA16.M, and CON22.M are similar in construction, and there are several things worth noting. Looking at the listings: the **h** and **g** vectors are specified, with elements depending on the coefficients. As described in Chapter III, zero valued coefficients are included, so the main tap elements in the **h** and **g** vectors are in the same positions.

**pad** is set equal to the number of coefficients (including zeros) up to, and including, the main taps (except in HAAR.M, where **pad** is not necessary). That is: **pad** = N - mt, where N is the number of coefficients and mt is the position of the first main tap. The file input, **c0**, is augmented by adding **pad** zeros onto the end of the vector. **c0** is then filtered using a built in Matlab function, **filter**, designed to work like a finite impulse response (FIR) filter. This function assumes the filter's delay chain is filled with zeros before the input is added. By adding **pad** zeros to **c0**, we allow the last of the non-zero elements of the input to reach the filter's main taps.

**i** is used both to eliminate the initial values of the filters' outputs, and to accomplish the decimation by 2. The first output value saved is the one whose number equals **pad**, and this corresponds to the first time inputs to the filter have reached the main taps.

TSINC.M uses N coefficients generated by TSINC_SU.M (also in Appendix A). This file generates the coefficients described in (3.11) for a modified sinc filter with a Hamming window. For $16 \le N \le 32768$, TSINC_SU.M uses linear interpolation to determine values for C and S from those listed in Table I. Once generated, the coefficients are saved on the hard drive, and called upon as needed by TSINC.M. The construction of TSINC.M is similar to the files described above, except, because the modified sinc filters are symmetric, no zero delays are necessary to align the main taps.

QMF.M was tested in several different ways with the HAAR.M, DA4.M, DA16.M, CON22.M, and TSINC.M files, and the code for all of these tests are listed in Appendix A.

**Tone Test**

The first of these were a suite of tone tests. In these, a sampled single frequency input was applied to the filter bank, and the outputs from various layers were examined. The purposes were, first, to verify the code for QMF.M is correct, and, second, to provide frequency response information to compare filters.

In the first of these tests, TTGENDAT.M and TONE.M were used to generate a 0.25 Hz sine wave with an amplitude of unity and **theta** (phase) of zero. The Haar filter was then used in the decomposition, and the results were displayed using DISCON.M and CROSST.M. DISCON.M displays a time frequency contour plot of the energy for a particular layer of the QMF output, and Figure 4.8 shows the results for layer seven. This is what we expect to see, and so it tells us QMF.M is working as expected, at least in this respect.

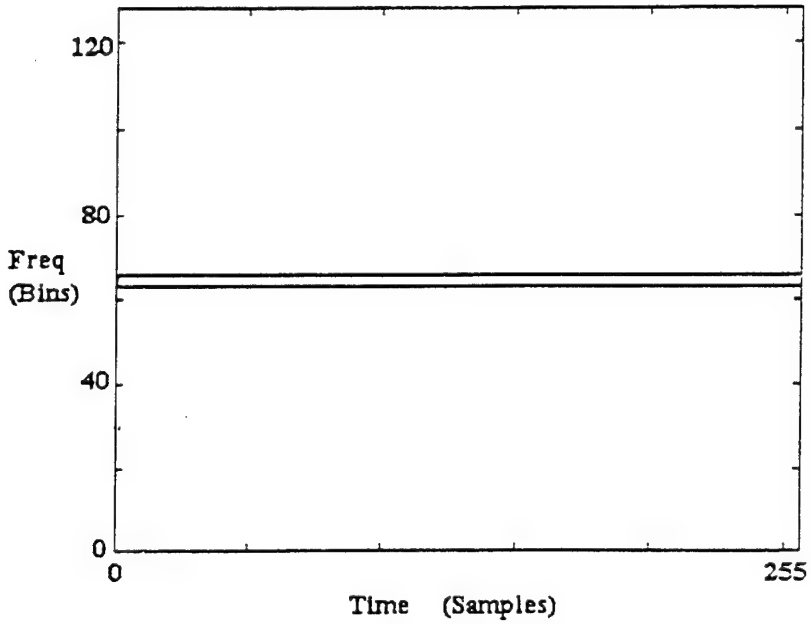



Figure 4.8. Layer Seven Time Frequency Diagram, 0.5 Hz Tone, Haar Filter

It is also convenient to look at the frequency for a given time sample. CROSST.M does this for a time sample approximately in the middle of the observation period. For the 0.25 Hz tone, these are shown in Figures 4.9-4.11 for layers two, seven, and 13, respectively. Comparing these, we see how the resolution improves as later layers are considered.

We should note that the symmetry in Figures 4.9-4.11 is due both to the fact the input signal's frequency is 0.25 Hz, and **theta** is zero. When **theta** is not zero, the amount of energy at

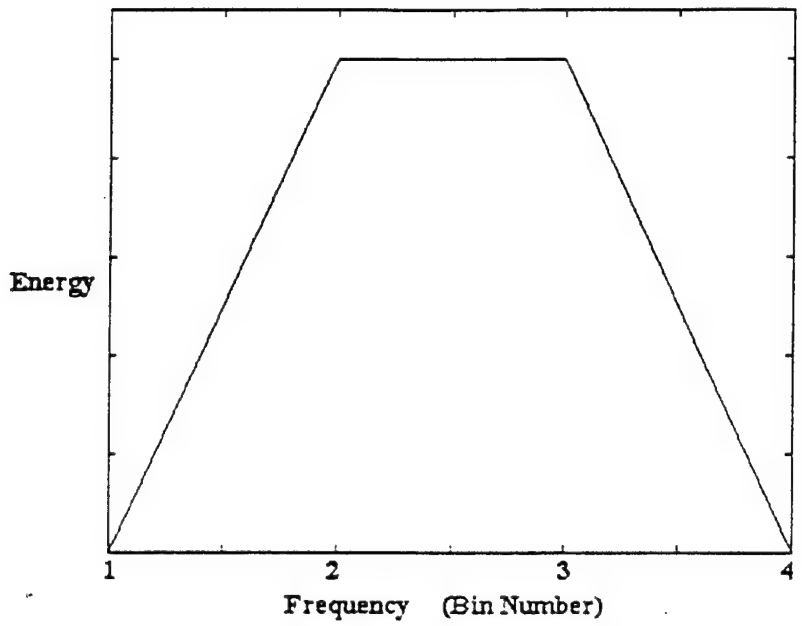



Figure 4.9. Layer Two, 0.25 Hz Tone, Haar Filter

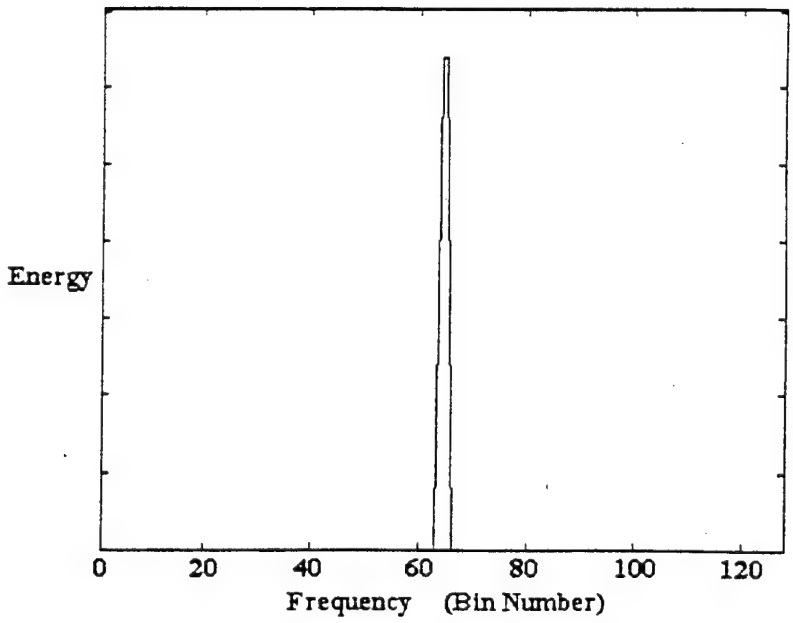



Figure 4.10. Layer Seven, 0.25 Hz Tone, Haar Filter

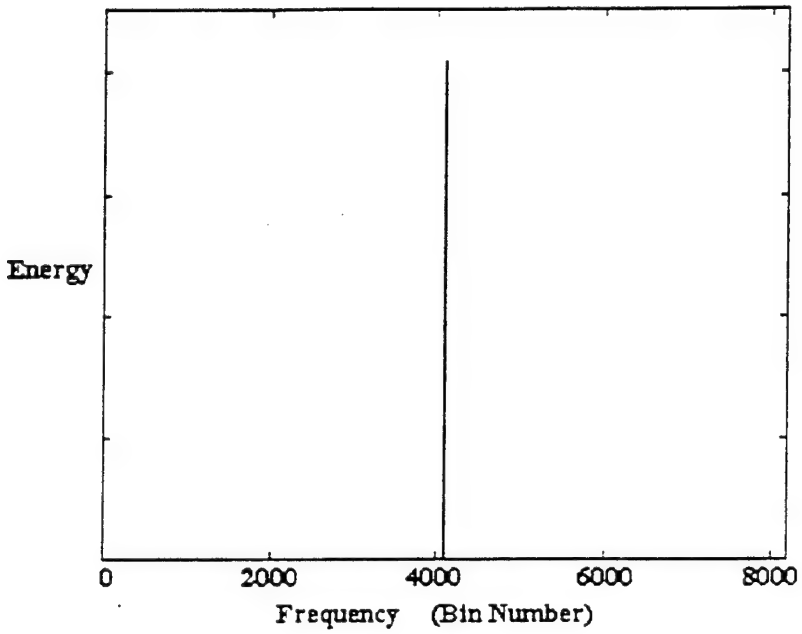



Figure 4.11. Layer 13, 0.25 Hz Tone, Haar Filter

each time remains constant, but more energy will occur in one or the other of the two bins adjacent to 0.25 Hz.

When the frequency is not the inverse of a power of two, a more serious effect is seen. This effect, called "leakage" and described in Chapter II, is shown in Figures 4.12-4.16 for representative time cross sections, when the input frequency is 0.3 Hz. This leakage is due to out of band energy passed by the filters used in QMF.M, and is evidenced here by the spikes at frequencies other than 0.3 Hz. The figures show the differences between the various filters. As we should expect, the modified sinc filter works best, while the 22 coefficient energy concentration filter does a better job than the Haar or Daubechies filters.

**Impulse Test**

QMF.M was also tested by inputting a sequence of all zeros, except for a single one near the middle of the sequence. This, again, was to verify the code is correct, and to provide a comparison of the time response of the various filters. This test was accomplished using IMPTEST.M.

Figures 4.17-4.22 show the results of IMPTEST.M. The figures show the squared coefficients of a particular layer of the QMF bank plotted along the time dimension. Rather than simply plotting a cross section at a particular frequency, all frequencies are shown to illustrate the spreading due to the various branches of the tree. The script file used to make the plots is called CROSSF.M.

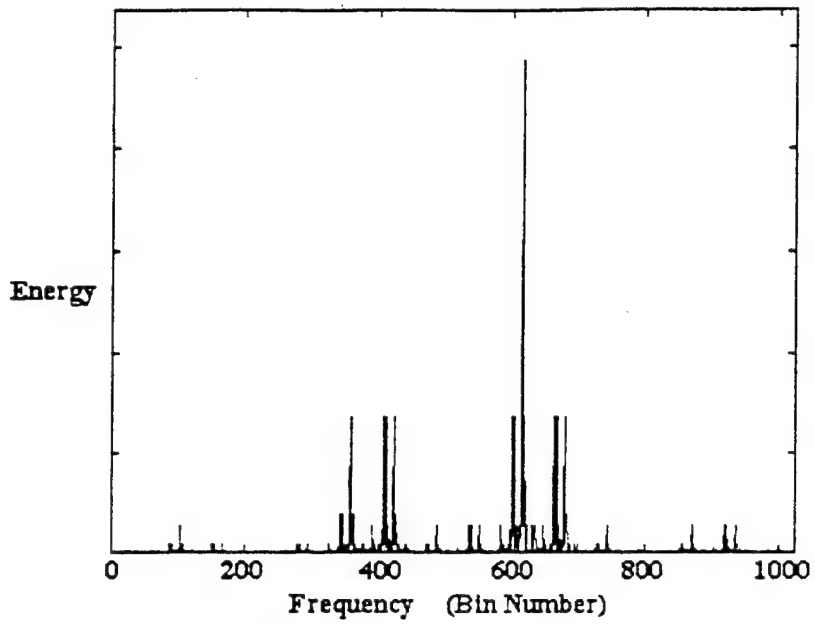



Figure 4.12. Layer Ten, 0.3 Hz Tone, Haar Filter

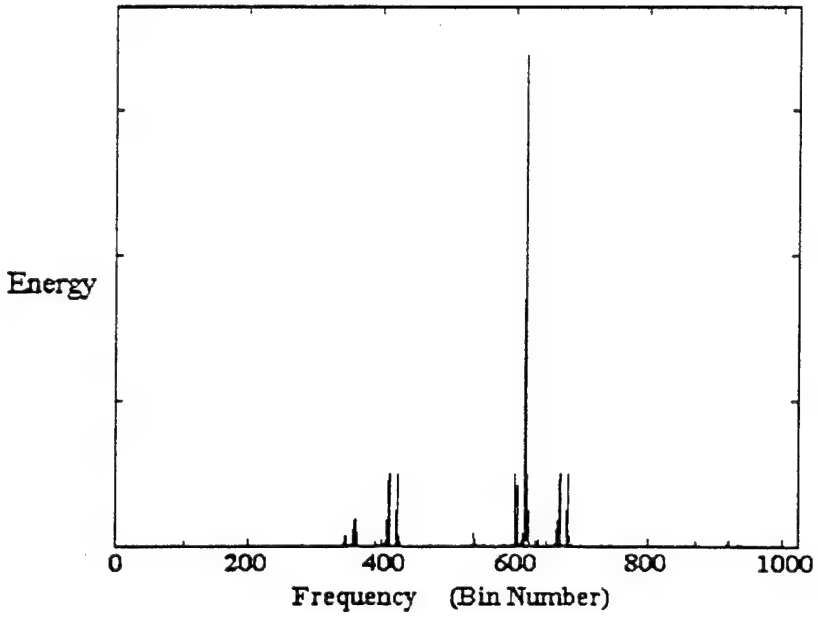



Figure 4.13. Layer Ten, 0.3 Hz Tone, Daubechies Four Coefficient Filter

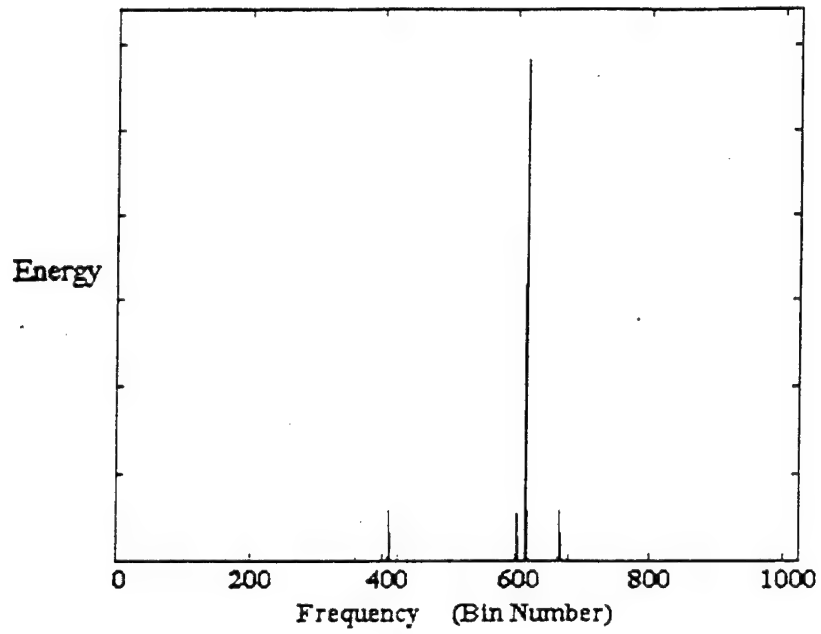



Figure 4.14. Layer Ten, 0.3 Hz Tone, Daubechies 16 Coefficient Filter

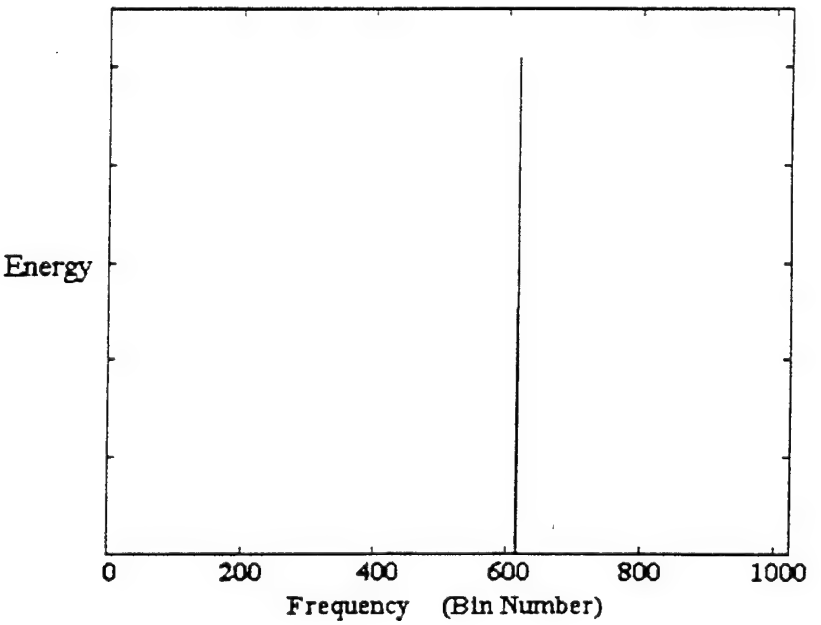



Figure 4.15. Layer Ten, 0.3 Hz Tone, Modified Sinc Filter

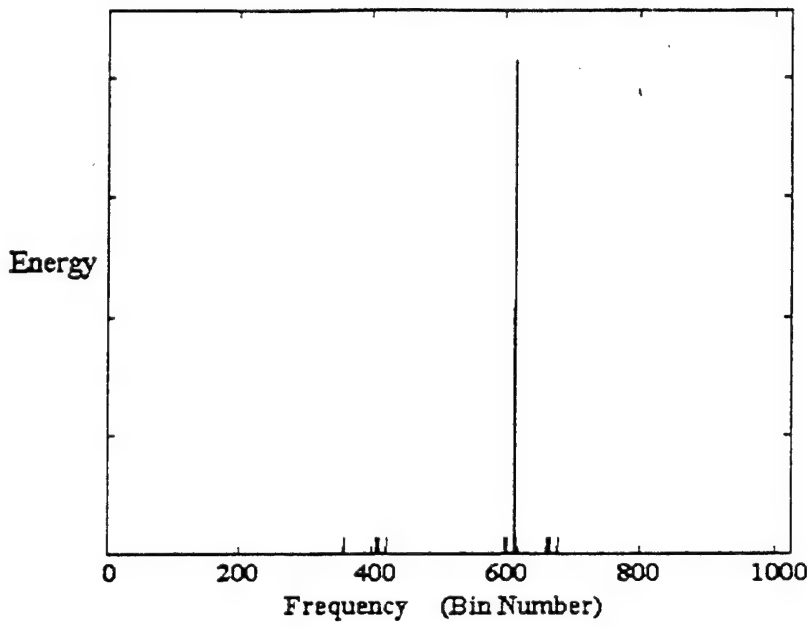



Figure 4.16. Layer Ten, 0.3 Hz Tone, Energy Concentration Filter

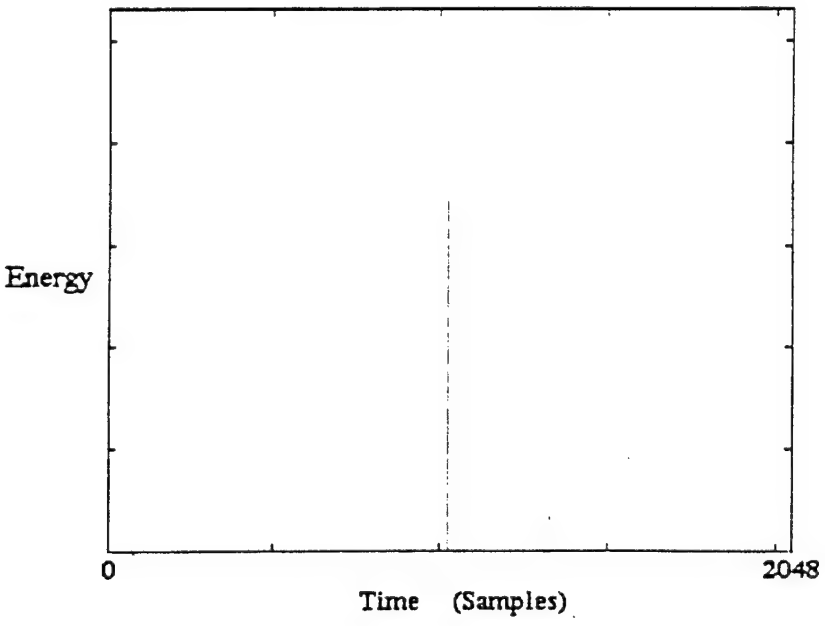



Figure 4.17. Layer Four, Impulse, Daubechies 16 Coefficient Filter

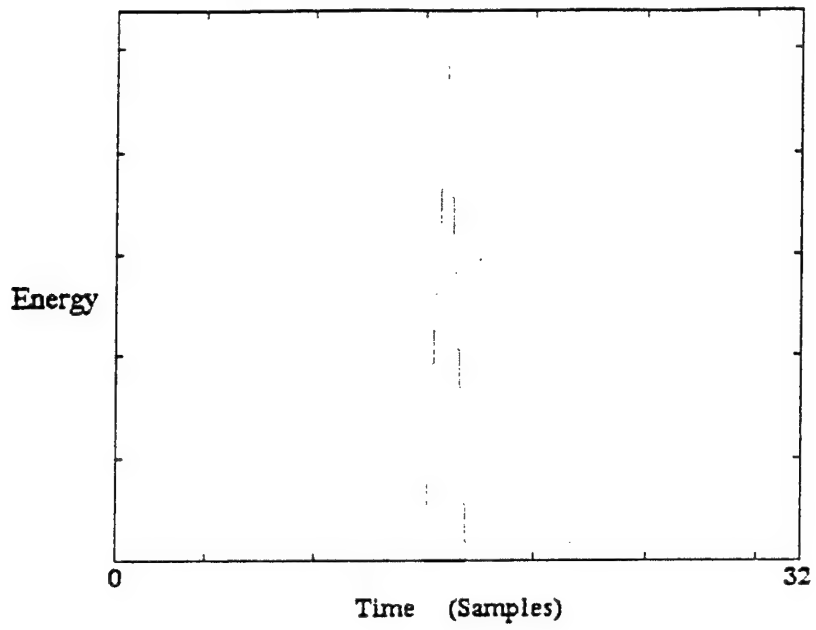



Figure 4.18. Layer Ten, Impulse, Haar Filter

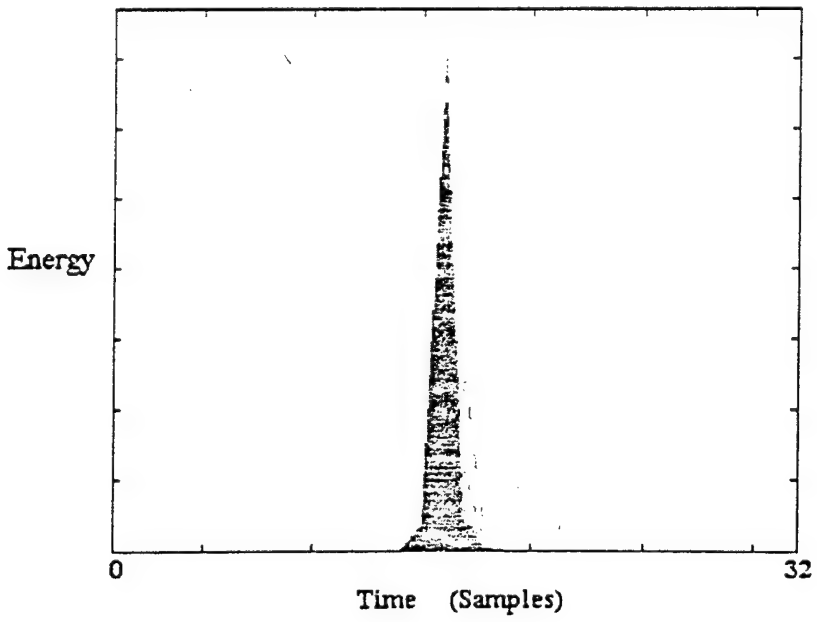



Figure 4.19. Layer Ten, Impulse, Daubechies Four Coefficient Filter

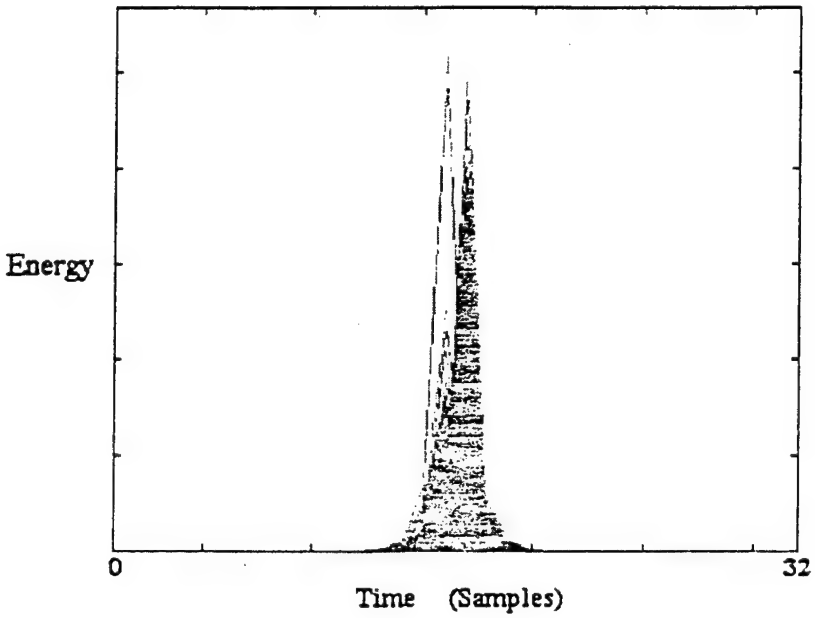



Figure 4.20. Layer Ten, Impulse, Daubechies 16 Coefficient Filter

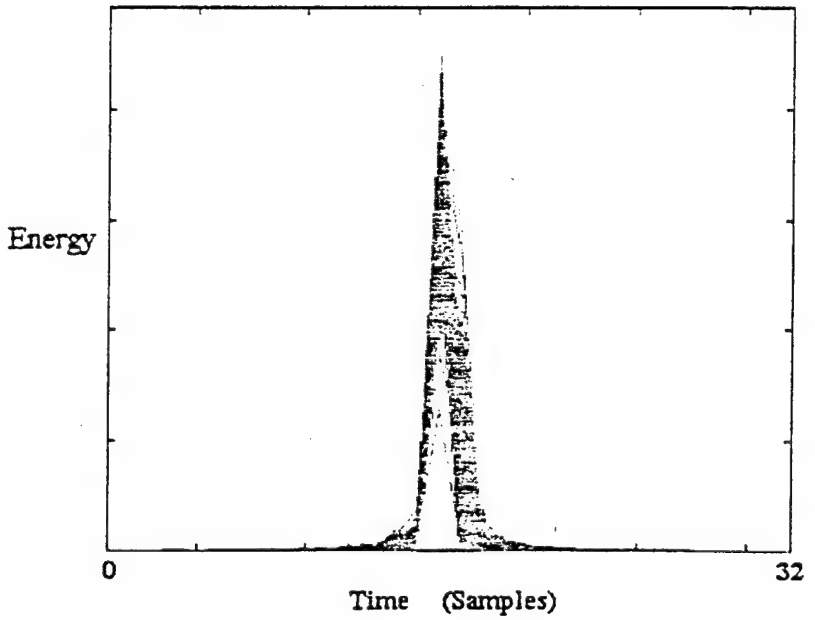



Figure 4.21. Layer Ten, Impulse, Modified Sinc Filter

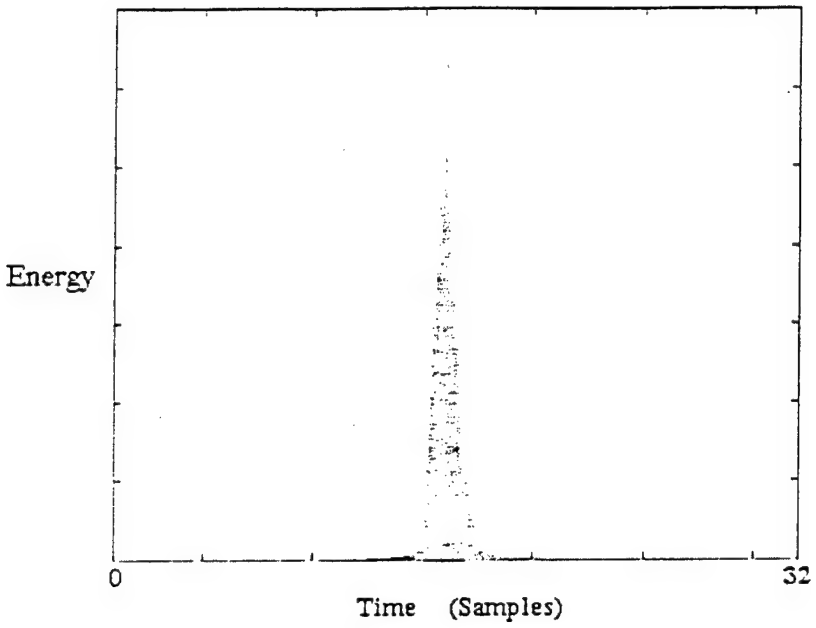



Figure 4.22. Layer Ten, Impulse, Energy Concentration Filter

Figure 4.17 shows layer four with the Daubechies 16 coefficient filter. We see the results are close to an impulse at the proper location, although some spreading, and differences in amplitude at different frequencies, are observed. This is what we expect, and it gives further indication QMF.M is working correctly.

To compare filters, Figures 4.18–4.22 show the results of IMPTEST.M at layer Ten. As we can see, the Haar filter yields the best results: only one point in time is non-zero, and that point has the same amount of energy for all of the branches of the filter tree. (This is indicated by the appearance of a line, rather than the gray region as in the other figures, which are actually many separate line plots of the outputs from the various branches.) We can see the 22 coefficient energy concentration filter also does a good job preventing the impulse energy from spreading, although the amount of energy at each branch varies.

### Noise Test

In this test, NOISTEST.M generates a sequence from NOISEGEN.M and inputs it to QMF.M. There are several things examined with this test.

First, the energy in the input sequence and energy at the output of the first layer are compared. They should be almost equal, although some (very small) round-off error can be expected. Also, for all filters except the Haar, there will be a small loss of energy from the output sequence because we only consider the portion generated while the non-zero elements of input sequence are passing the filter's main taps. (In other words, some energy from the beginning and end of the input sequence will be lost.)

Second, the distributions of the input sequence and first layer output are tested to determine whether they are Gaussian with a mean of zero and variance of one. We expect this from the input since that is what NOISEGEN.M is supposed to generate. Since the filtering process is linear, the output should also be Gaussian, and, of course, no DC energy should be added. Since the variance represents the noise energy and, from the test above, we know the energy from the input sequence is equal to the energy of the output sequence (except for energy lost at the beginning and end of the input sequence) the variance of the output should equal the variance of the input: one.

The core of this test is NORMGOF.M, which takes an input sequence and applies a Chi-Squared Goodness of fit test [14] [29] [40] to determine whether or not the sequence is likely to have a normalized Gaussian distribution (mean of zero and variance of one). The program separates the input data into eight bins, set up to contain approximately equal numbers if the input has a normalized Gaussian distribution. The test variable, v, is then computed by comparing the binning results to the expected results. v is then compared to various percentage points on the Chi-Squared distribution to determine the degree of confidence with which we can say the input came from a normalized Gaussian distribution. Since the program uses eight bins, and we know the mean and variance, the program uses the Chi-Squared distribution with seven degrees of freedom.

The actual test was conducted three times for each of the filters. The seeds used for the three tests were: 17, 1489, and 8016. The ratio

$$\frac{\text{energy in - energy out}}{\text{energy in}} \tag{4.2}$$

was on the order of $10^{-14}$ for the Haar filter, and reached a maximum of $1\times10^{-3}$ for the Daubechies 16 coefficient filter. For the second part of the test, our null hypothesis is that the data has the normalized Gaussian distribution, and we set the confidence interval at 0.05. (So there is a 5% chance of rejecting the null hypothesis when it is true.) The filters passed the test in every case, indicating the process itself is linear and that the filter coefficients do not affect the mean or variance. (The input sequence for the 8016 seed failed at the 5% level, by the way. It did, however, pass at the 2.5% level. Even in this case, the first layer output for each of the filters passed at 5%.)

## Summary

This chapter discussed the portions of Matlab code used to generate input waveforms and decompose them using a QMF bank tree. Tests for the tree were discussed. These were performed partly to verify the files work correctly and partly to provide relative comparisons between QMF filters.

Specifically, the tone tests show the QMF bank tree decomposes tiles correctly in the frequency dimension and the impulse test shows the same in the time dimension. Both tests show the leakage and dispersion of signal energy due to the non-ideal characteristics of the filters. The noise test was used to verify linearity, to show the filters are not biasing the data, and to show energy is conserved from layer to layer (except at the beginning and end of data sequences).

## V. Interception of Fast Frequency Hop (FH), Time Hop (TH), and FH/TH Signals

### Introduction

Here, we look into the detection and feature extraction of spread spectrum signals whose cells have time bandwidth products of unity. These include the Fast FH, TH, and FH/TH signals.

In this paper, "detection" refers to the process, by the interceptor, of determining whether spread spectrum signals are present. It is a binary decision: either the interceptor decides no signals are present, or decides one or more signals are present. We do not necessarily obtain any information about how many signals are present, or the characteristics of the signal(s).

"Feature extraction," on the other hand, will refer here to finding key cell characteristics that can be used by a classifier to determine how many signals are present and each of those signals' key features. These characteristics include estimates of the SNR of each cell at the interceptor, and estimates of the cell dimensions and location in the time frequency plane.

We will begin this chapter with a quick discussion of the characteristics of spread spectrum cells with time bandwidth products of unity. We will then shift emphasis and examine the radiometer--the detection receiver to be used when practically nothing (including the time bandwidth product) is known about the structure of a signal to be intercepted. The radiometer will be shown to be the optimal receiver in this case, and its implementation with the QMF bank tree will be discussed. Since the cell time bandwidth of unity for the signals examined in this chapter represents a significant increase in our knowledge about the signals' structure, our goal will be to find a detection algorithm that takes advantage of this to achieve better results than the radiometer.

Our approach in this search will be to first consider detection when the interceptor knows most of the signal characteristics. This problem has been solved and is widely discussed in the literature. We will see how the QMF bank tree can be used to implement the solution. We will then generalize the problem by assuming that the interceptor knows less about the signal to be detected, and show how to generalize the detection algorithm to handle this. The algorithm we develop will then be extended to obtain the characteristics needed for feature extraction.

### Spread Spectrum Signals With Cell Time Bandwidth Products of One

Before discussing detection, we should look at these spread spectrum signals in some detail. Figure 5.1 shows a portion of a signal containing a cell with a length of T and amplitude of A. (In the case of the figure, it must be a TH or FH/TH signal, because there are no cells immediately adjacent to the one shown.) Because the hop rate is greater than the information rate, the cell will be a portion of a sinusoid that does not change frequency. The frequency of the sinusoid is the cell's center frequency (the "channel frequency"). The phase of the sinusoid at the beginning of the cell is, of course, determined by the transmitter, and to the interceptor is considered to be random. The analog cell energy is

$$\varepsilon = \frac{A^2 T}{2} \tag{5.1}$$

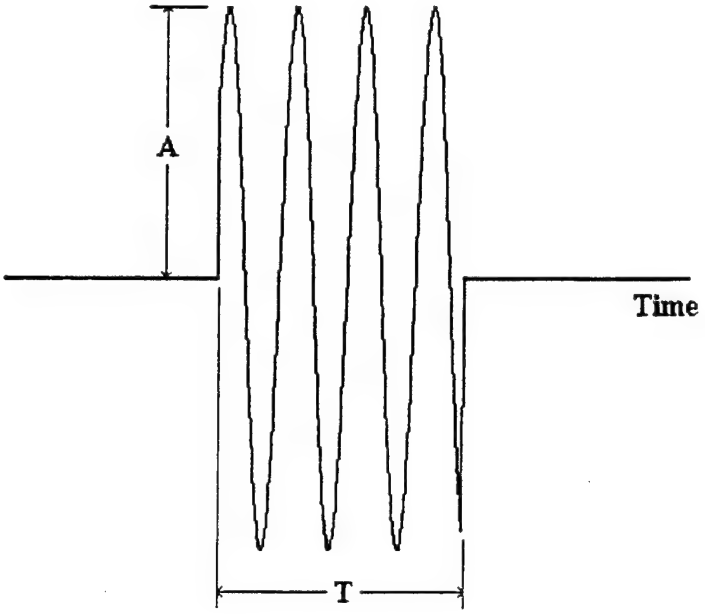



Figure 5.1. A TH or FH/TH Cell

Our interceptor samples this analog signal, and when we take the sum of the squares of the samples, we will have an approximation of the cell energy. However, because the sampling time and sinusoid phase are not synchronized, we cannot know exactly how the energy will be distributed among the individual samples.

To see what the energy distribution of a cell looks like in the frequency domain, we note the signal in Figure 5.1 can be described mathematically as a sinusoid of infinite duration multiplied by a rectangle function. Using Fourier Transforms, this gives us a sinc function, (2.7), in the frequency domain, centered at the sinusoid's frequency, $f_c$, and with nulls at integer multiples of 1/T from $f_c$ [40]. The energy distribution is, therefore, the area under a sinc squared function, and this is shown in Figure 5.2, with the cell energy normalized to one.

The figure shows how the energy is distributed in the frequency domain. Taking the cell bandwidth as 1/T gives us B in the figure, the region under the main lobe covering 0.774 of the cell's energy. The amounts of energy distributed in sidelobes, and in other portions of the main lobe, are also shown.

Since we are sampling an analog signal, we should note that there will always be some aliasing present in our sampled signal, since the sinc's sidelobes extend infinitely. As the figure shows, however, the amount of energy in the sidelobes drops off dramatically. As we mentioned in Chapter IV, by restricting the signal in our simulations to the range [0.125, 0.375] Hz (normalized), we will effectively eliminate the aliasing.

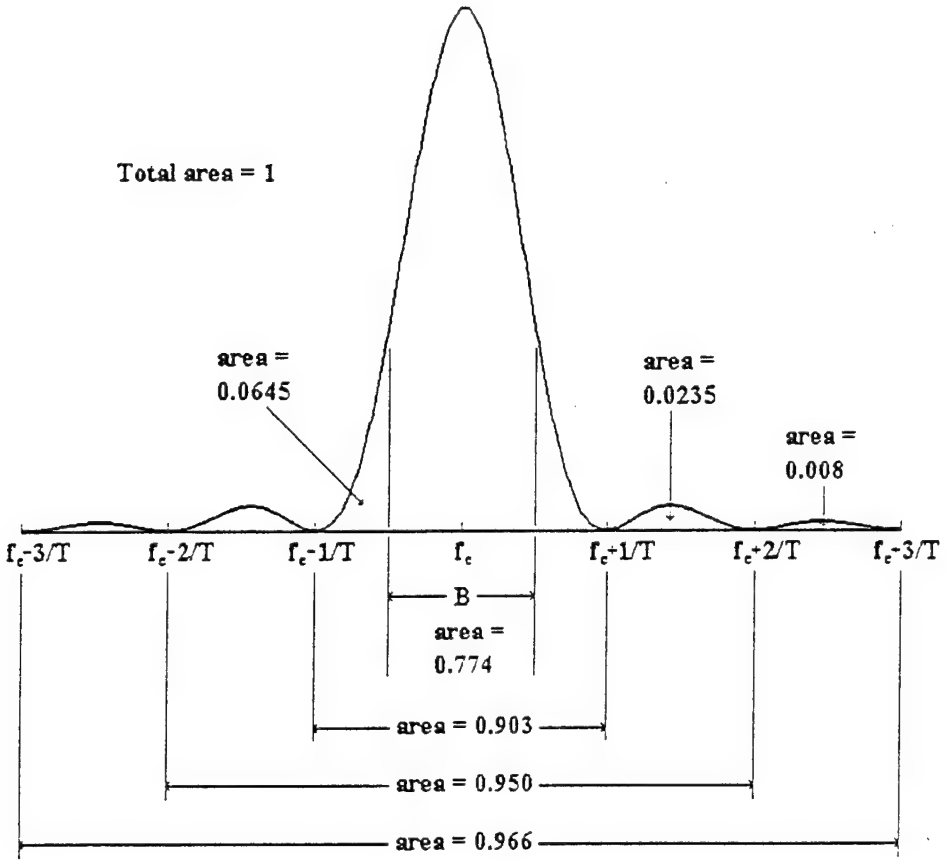



Figure 5.2. The Sinc Squared Function, With the Areas Under Key Portions of the Curve

We can now envision how a spread spectrum cell might appear when analyzed with a QMF bank tree. Figure 5.3 shows this. Figure 5.3a shows the time frequency plane, so we are looking down on the cell, which has sharp edges in time, and sidelobes in the frequency dimension. The time bandwidth product of the cell, taking our definition of the bandwidth as B = 1/T, will be one. Figure 5.3b shows this same cell, with a possible tiling overlaid. As we saw in Chapters II and III, if we could have ideal QMF bank filters, the square of each element out of the bank would represent the signal energy in a respective tile. Since each of the tiles has a time frequency product of 0.5, it will take at least two to cover the cell, not counting the edges of the cell's main lobe or the sidelobes.

In the figure, we show the more general case, in which the tiles and the cell are not synchronized, and the cell's dimensions are not integer multiples of the tiles'. Of course, the tiles' dimensions change by a factor of two depending on the layer of the tree used.

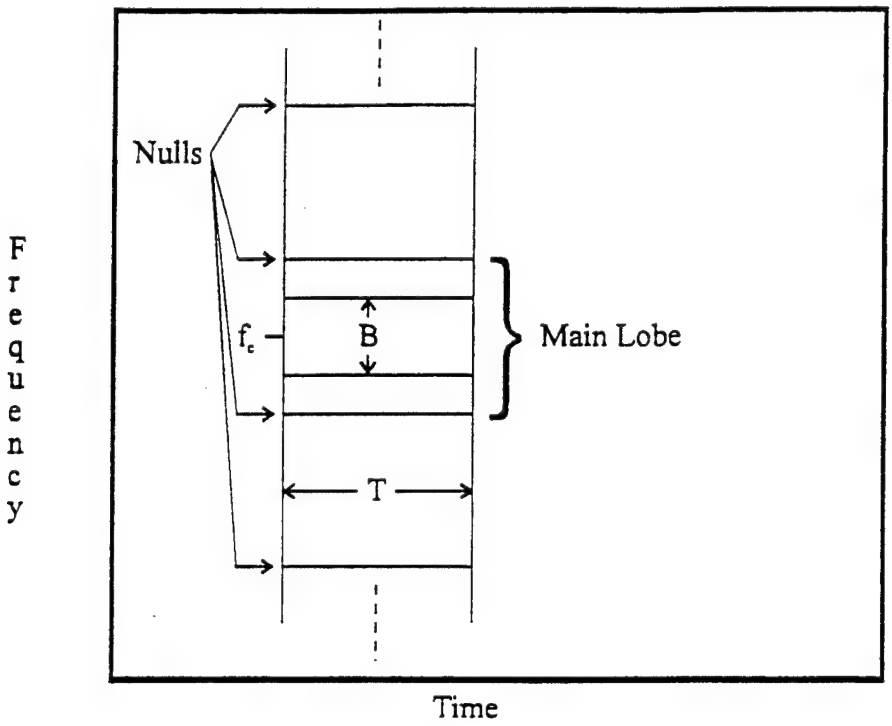



Figure 5.3a. Spread Spectrum Cell in Time Frequency Diagram

## Detection

### Radiometer

One of the basic building blocks of energy detectors described in the literature is the radiometer [15] [16] [18] [19] [41] [45]. Figure 5.4 shows a block diagram for the radiometer. The initial bandpass filter has a bandwidth, W. The passed signal is squared, and the results are collected for a time slot, T, by the integrator. The output is the energy of the input signal in the rectangle of the time frequency plane described by T and W. (This is, of course, actually an approximation, since we are assuming an ideal bandpass filter.)

It is easy to see the radiometer can be implemented by the QMF bank tree, by adding the squared coefficients of the 2TW tiles covering the area described by T and W. When the radiometer input is white Gaussian noise (WGN), the radiometer output can be shown to have a Chi-Square pdf, with 2TW degrees of freedom [41]. This agrees with Equation (2.19) for the sum of squared coefficients. Likewise, when the input consists of a signal and WGN, the radiometer output will have a Chi-Square pdf with 2TW degrees of freedom, and a non-centrality parameter equal to the signal energy, agreeing with (2.29).

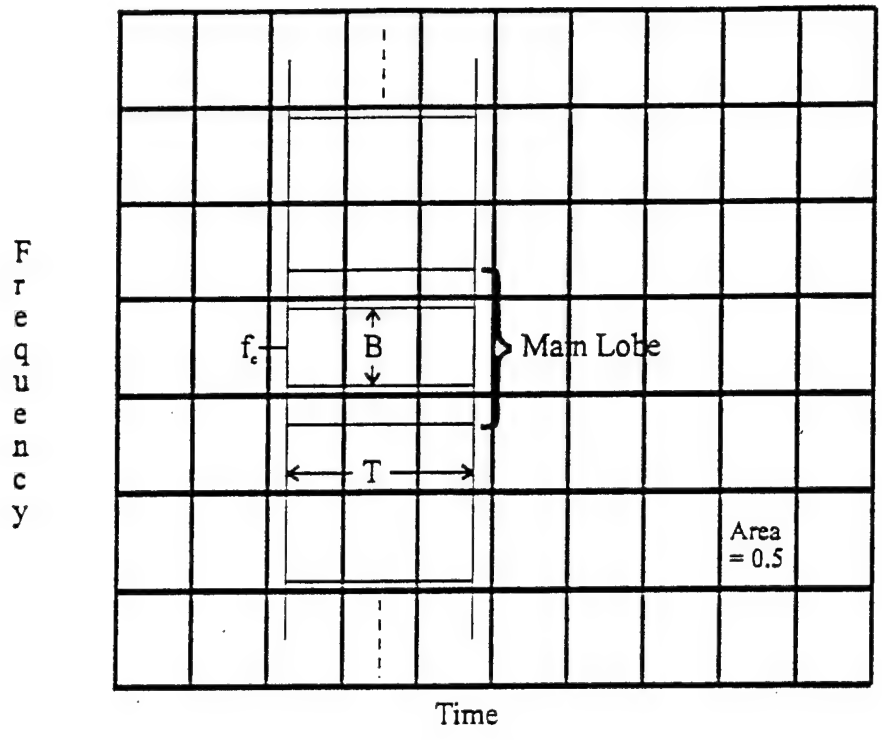



Figure 5.3b. Spread Spectrum Cell in Time Frequency Diagram

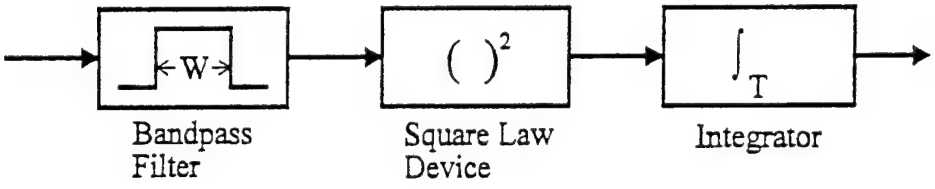



Figure 5.4. Radiometer

We are now in a position to show that if we know nothing about a signal we're trying to detect, except for the dimensions of the time frequency plane in which it is located, the radiometer set to match these dimensions is as good as any detector we can build with the QMF bank tree. (We use the QMF bank tree for the sake of a specific example because that is the method of decomposition we are concerned with in this document. However, the following analysis applies to any linear orthogonal decomposition of the input waveform.)

Suppose we are using a tree with orthogonal filters to cover the dimensions of interest in the time frequency plane, and we are looking at the output from a certain layer. Since we are collecting a data point for each tile, we will have lots of data which can be evaluated in a number of ways. For example, we can:

1) Compare the energy in each tile, or in groups of adjacent tiles, against a threshold, and if a certain number exceed the threshold, decide a signal is present.

2) Find the mean, variance, or some other moment and compare this to a threshold.

To consider this situation, we normalize the signal and noise energy of the input so the noise variance equals one, and consider the pdf of each coefficient output for each tile in the examined layer. Let $a_k$ be the coefficient for the k-th tile. $a_k$ will have the distribution

(5.2) $$a_k \sim N(0,1)$$

when noise only is present in the tile (where we use N(m,v) to indicate a Gaussian distribution with a mean of m and variance of v), and

(5.3) $$a_k \sim N(s_k,1)$$

when $s_k^2$ of the signal's energy is present in the tile. We assume the signal has no DC component, so

(5.4) $$E[s_k] = 0$$

Now let $p_n$ be the (unknown) probability that a tile contains no signal, and $p_{s_k}$ be the (unknown) probability that a tile contains the signal energy $s_k^2$. We will then have an overall distribution for an arbitrary tile, $a_i$, of

(5.5) $$a_i \sim p_n N(0,1) + \sum_{s_k} p_{s_k} N(s_k,1)$$

where

(5.6) $$p_n + \sum_{s_k} p_{s_k} = 1 \qquad \text{and} \qquad p_n \neq 1$$

Of course, we know practically nothing about the distribution of $s_k$ or about $p_n$ or $p_{s_k}$, but, because we are dealing with Gaussian distributions, we can see (5.4) and (5.5) give us

(5.7) $$a_i \sim N(0,\delta^2) \qquad \text{where } \delta^2 > 1$$

for an arbitrary tile when a signal is present. Thus, when we are almost totally ignorant of the signal structure, we see the statistical variance is a sufficient statistic to test for detection. In other words, we will obtain no further information by measuring anything other than the variance in our samples of $a_i$. If this variance is close to one, we should decide no signal is present; but if it is significantly greater than one we should decide a signal is present. Going back to the question of the best way to

analyze the data from the output of the tree, 2) should be our choice. We should find the variance of the output, and compare this to a threshold.

Now, we note

$$E\left[a_i^2\right] = \delta^2 + \left\{E\left[a_i\right]\right\}^2 = \delta^2 \tag{5.8}$$

where we recognize the left side is the mean of the energy in our tiles. Therefore, instead of working with the variance of the coefficients, it is mathematically equivalent for us to work with the mean of the energy of the coefficients. But the statistical mean is (assuming ideal tiles)

$$\frac{1}{2TW}\sum_{i=1}^{2TW} a_i^2 \tag{5.9}$$

where T and W are the dimensions of the observed time frequency plane. We see (5.9) is proportional to the total energy in the region described by T and W, and, therefore, the best we can do with the QMF filter tree is only as good as a radiometer set to T and W.

### Optimal Detector

In the case described above, we assumed nothing about the signal to be detected except that it is contained in the portion of the time frequency plane described by T and W. For the signals in this chapter, however, we have one other important piece of information: We know they consist of cells with time bandwidth products of one. Our goal, then, ought to be to find a way to take advantage of this to design a detector that will yield a better performance for these signals than the radiometer.

We will now discuss intercept receivers that have been developed for cases where the interceptor has a lot of knowledge about the signal to be detected. These receivers, the "optimal detector" and "filter bank combiner" (FBC) have been widely presented in the literature [15] [16] [18] [19] [30] [35] [41] [42] [45]. Our purpose for discussing them here is to: 1) show how the QMF bank tree can be used as an FBC, and 2) present the interceptor's best case condition. That is, the one where the interceptor knows the signal's channelization, hop rate, timing synchronization, and, for the optimal detector, the amount of cell energy received. Later in the chapter, we will relax the requirements and see how we can modify the intercept receiver (using the QMF bank tree) and develop a detection algorithm for cases where the only thing known is that the cells' time bandwidth products are one.

For the Fast FH signal, in the particular case when the interceptor knows the signal features listed above, the "optimal" detector is the one shown in Figure 5.5 [16]. This receiver consists of a bank of radiometers, with each one tuned to a particular hop channel, and with the integration set to cover a hop period (synchronized with the hop). In this way, each of the radiometer outputs, $x_1$ to $x_M$, will consist of noise energy, and most of the cell energy if a signal cell is present. (If we assume the bandpass filters in the radiometers have perfectly sharp cutoffs, and have a bandwidth of B, where B is as described in Figure 5.2, the radiometer will pick up 0.774 of the cell's energy, for a loss of 1.1 dB over a filter with a matched magnitude response.)

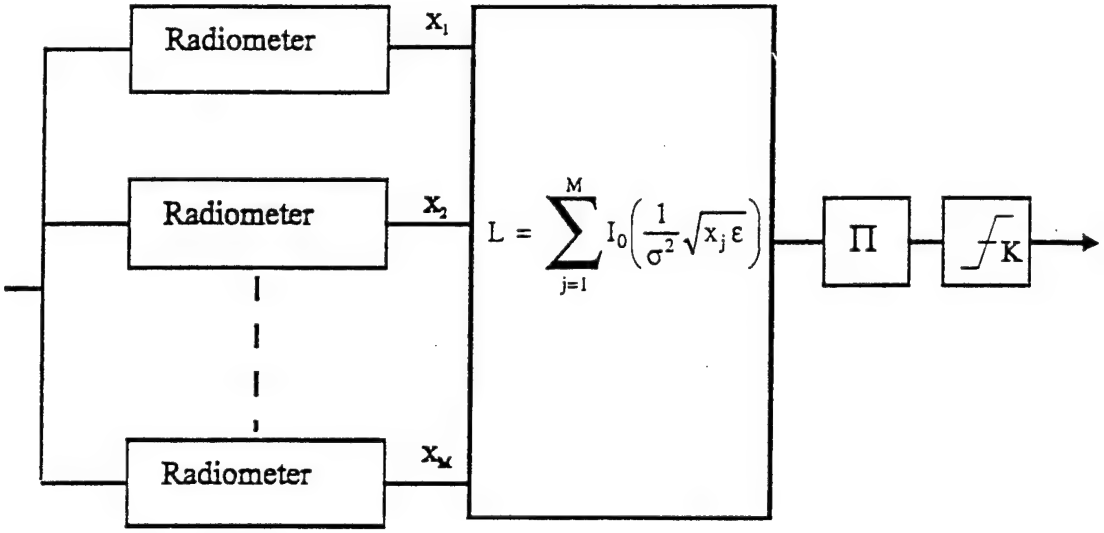



Figure 5.5. Optimal Detector For Fast FH Signals

Since, for the Fast FH signal structure, there will only be one signal cell at any given time, the radiometer outputs are not statistically independent when a signal is present. Using likelihood ratio arguments, the optimal test statistic for each time slot is [16]

$$L = \sum_{j=1}^{M} I_0\left(\frac{1}{\sigma^2}\sqrt{x_j \varepsilon}\right) \tag{5.10}$$

where

| | |
|---|---|
| $I_0(\bullet)$ | is the modified Bessel function of the first kind, zero order |
| $\varepsilon$ | is the cell energy |
| $\sigma^2$ | is the noise variance |
| M | is the number of Fast FH channels |

The values of L for each time slot are saved and multiplied together for the observation time. This number is then compared to a threshold (K in the figure) to decide if a signal is present.

Although shown for the Fast FH signal, modifying Figure 5.5 for the TH or FH/TH signals is straightforward. For the TH signal, the radiometers are adjusted to collect energy for each possible time slot for each hop. For the FH/TH signal, a radiometer is assigned to each possible time/frequency slot for each hop.

**Filter Bank Combiner (FBC)**

Perhaps the most unreasonable assumption for the optimal detector is that the interceptor knows the signal's cell energy. A simplification that obviates the need for this information, and also makes implementation easier, is the FBC, shown in Figure 5.6. This receiver retains the radiometer bank from the optimal detector, but compares each radiometer output against a threshold, treating

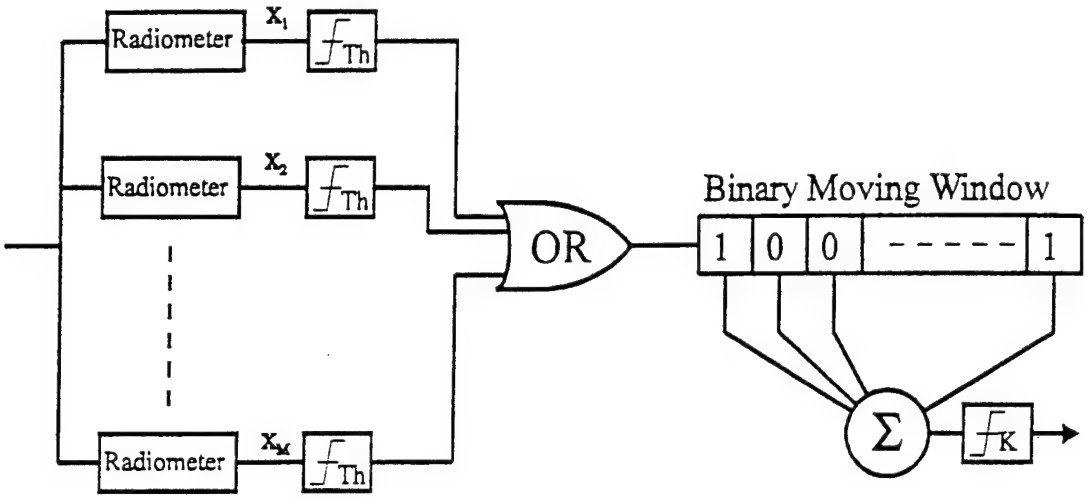



Figure 5.6. Filter Bank Combiner

each independently from the others. The results, which can be represented at each threshold's output as a binary "1" if a cell is determined to be present, and "0" if absent, are then fed into a logic OR gate and an overall cell/no cell decision is made for each time slot. There are a number of different possible ways to combine the decisions of each of the time slots. The one shown in the figure, and the one we will consider, is the binary moving window (BMW). In the BMW, we collect the binary decisions for the $N_p$ time slots in the observation period, and the number of "detections" are compared to a threshold, K. When the number exceeds this threshold, a signal is taken to be present.

The analysis of the FBC is straightforward. As usual, we normalize the noise variance to one. Since each radiometer covers a time bandwidth product of one, the pdf of the outputs will be Chi Squared with two degrees of freedom

$$f_n(x) = \frac{1}{2}\exp\left(-\frac{x}{2}\right) \tag{5.11}$$

when noise only is present, and Chi-Squared with two degrees of freedom and a non-centrality parameter

$$f_s(x) = \frac{1}{2}\exp\left(-\frac{\varepsilon_c + x}{2}\right) I_0\left(\sqrt{\varepsilon_c x}\right) \tag{5.12}$$

when the portion of a cell's energy collected by the radiometer, $\varepsilon_c$, is present. These are the pdfs shown in Figure 5.7. When the threshold, Th, is set, as shown in the figure, the probability of cell detection, $Q_d$, given that one is present, is the area under $f_s(x)$ to the right of the threshold. Likewise, the probability of false alarm on the channel, $Q_{fa}$, given no cell is present, is the area under $f_n(x)$ to the right of the threshold.

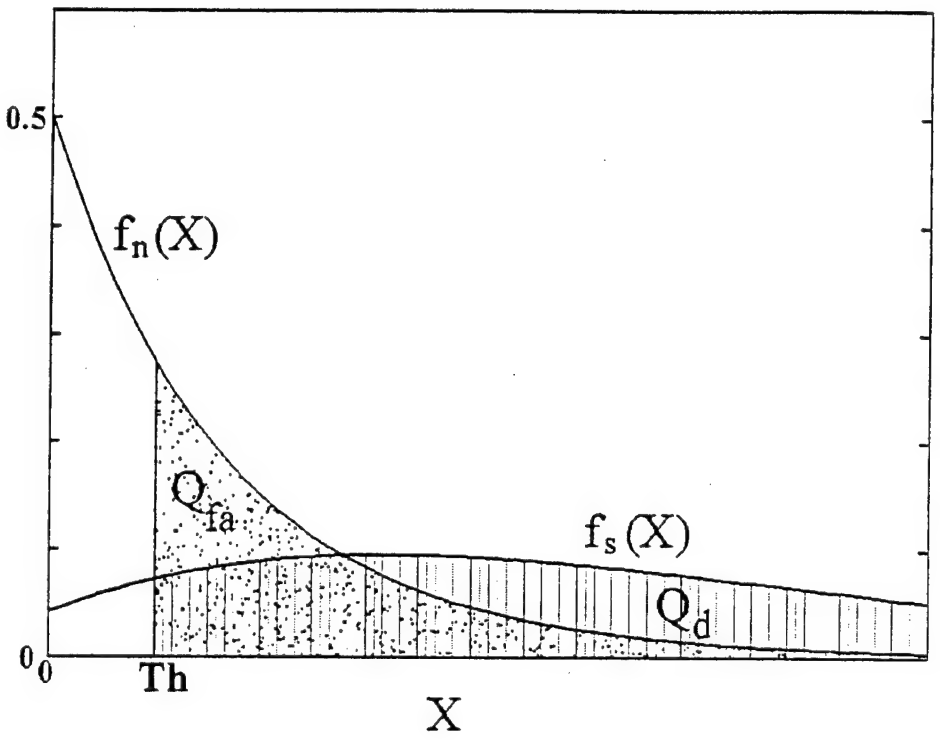



Figure 5.7. Chi Square Probability Distribution Functions

To compute the probability of detection, $p_d$, and probability of false alarm, $p_{fa}$, at the output of the OR gate, the following equations are used [16]

(5.13) $$p_{fa} = 1-(1-Q_{fa})^M$$

(5.14) $$p_d = 1-(1-Q_d)(1-Q_{fa})^{M-1}$$

where, for (5.14), we assume only one signal is present and, therefore, only one Fast FH cell will be present in a time slot.

With the BMW, the overall probability of false alarm, $P_{fa}$, is related to $p_{fa}$ by the cumulative binomial distribution [16]

(5.15) $$P_{fa} = \sum_{i=K}^{N_p} \binom{N_p}{i} p_{fa}^i (1-p_{fa})^{N_p-i}$$

and the overall probability of detection, $P_d$, will be related to $p_d$ in a like manner

$$P_d = \sum_{i=K}^{N_p} \binom{N_p}{i} p_d^i (1-p_d)^{N_p-i} \tag{5.16}$$

When a large number of time slots can be evaluated by the detector, the effect of the BMW is to "amplify" the difference between $p_d$ and $p_{fa}$ ($p_d$ will, of course, always be greater than $p_{fa}$), leading to a greater difference between $P_d$ and $P_{fa}$.

As with the optimal detector, the FBC shown here is designed to detect the Fast FH signal; but modifications to detect the TH or FH/TH signals are identical to those described above for the optimal detector.

**Implementation of the FBC With the QMF Bank Tree**

It is easy to see how we can implement the bank of radiometers in the FBC or optimal detector with the QMF bank tree. Since the interceptor knows the signal's channelization and hop timing, the sampling can be synchronized so the QMF tiling is aligned with the FH cells. Output can be taken from the layer of the tree where the tiles' frequency dimension is equal to the cells' bandwidth, and energy from adjacent time tile pairs can be added to yield the radiometer outputs.

A description and listing of the Matlab code to implement the FBC is given in Appendix B. One of the primary goals of this simulation is to determine how closely the QMF bank tree, with the filters we have described in Chapter III, come to matching the performance predicted by (5.11) - (5.16) for ideal tiling. We can measure this with receiver operating characteristic (ROC) curves, plotting the probability of detection against the probability of false alarm.

In order to estimate these values to an accuracy of about one decimal place in the simulation, we have made 100 runs with noise alone as an input (to determine $P_{fa}$) and 100 runs with a signal and the same noise (to determine $P_d$). For clarity, we will call these 200 runs a "set." By using a vector of thresholds, we obtain a vector each for $P_d$ and $P_{fa}$, which we use to plot the curve.

Unfortunately, it turns out that a Fast FH signal, with the cell energy and the detector's thresholds set so as to obtain both a $P_d$ and $P_{fa}$ in our range of interest, will yield poorer results than the same signal intercepted with a radiometer detector. With this signal, the FBC outperforms the radiometer in the region of the ROC curve where one has a very low probability of false alarm. These probabilities can be readily calculated with (5.11) - (5.16), but cannot be verified with simulation without an increase, by several orders of magnitude, in the number of runs performed in each set.

Despite this, sets of simulations were run with a Fast FH signal with a (normalized) cell length of 128 seconds, 32 channels, and cell energies of one and three (with a noise variance of one). In these cases, all of the different wavelet filters yielded similar results. As expected, the radiometer outperformed them all.

A FH/TH signal configured so there would be approximately ten cells in the observation period, and each cell would have an energy of 20, was then examined. Overall, there is less signal energy than in the Fast FH case, but it is better concentrated. Using (5.11) - (5.16) it can be verified the FBC, with the threshold K set equal to one, outperforms the radiometer in the region where $P_{fa}$ is

in the range of [0.1, 0.5]. By only considering K = 1, the simulation code can be used on a FH/TH signal (vice a fast FH signal) without modification. In this case, the BMW acts like another OR gate. We actually are looking at every region of the time/frequency plane that can contain a cell, and if the energy in any one of them exceeds the threshold, we declare a signal to be present.

The results of the simulations are shown in Figure 5.8. The modified sinc filter (with 512 coefficients), the 22 coefficient energy concentration filter both described in Chapter III, and the Haar filter, were all tried. For each of these, three sets of runs were made. Also plotted in Figure 5.8 are the theoretically expected results for perfect tiling. In that case, as we saw in Figure 5.2, only 0.774 of the total cell energy would fall in the tile, so $\varepsilon_c = 15.48$ was used in (5.12) to obtain the curve. As the figure shows, in this case the modified sinc filter did the best, followed by the energy concentration filter. The Haar filter did not do nearly as well.

### The Nine Tile Scheme

We now move onto cases that have not been discussed in the literature. First, we relax the requirement that the interceptor know the channelization and hop synchronization of the signal(s) to be detected. We will, for the present, assume the interceptor knows the hop rate to within a power of two (later we will relax this requirement).

The immediate result of this change is that the signal cells will no longer be synchronized with the tiles, and we will have the case depicted in Figure 5.9 (where, for clarity, the portion of the cell outside of its main bandwidth is not shown). For this figure we have picked the QMF band tree layer where the tiles' lengths in time are from 0.5 to one times the cells'. Since each layer's tiles are twice as long as the last layer's, this layer will always exist for any cell length. For clarity, we will call this the "$\beta$ layer" for a particular cell (or group of cells with the same length). (We will justify the use of this layer shortly.) Looking at the frequency dimension, a cell's $\beta$ layer tiles will have a height that ranges from B/2 to B.

#### Strategy

Our strategy is going to be to take adjacent tiles in the $\beta$ layer to form a "block," and to compare each block's energy against a threshold to decide whether the block contains a signal cell. To do this, we must decide how large the blocks should be. In our development, we will assume ideal tiling.

Obviously, we want a block large enough to cover most of the cell's energy. On the other hand, we do not want it so large that there is a high probability of accidentally picking up energy from more than one cell. We also want to keep the block as small as possible to reduce the amount of noise energy we pick up.

The first thing to notice about the $\beta$ layer is that at most, three columns of tiles will always cover a cell. Evidently, then, the block should consist of three tiles in the time dimension. The frequency dimension is more difficult. What we need to do is to figure out how much of a cell's energy would be likely to be covered with different sized blocks, take the noise energy into account, and, using Equations (2.19) and (2.29), find the best probability of detection vs probability of false alarm.

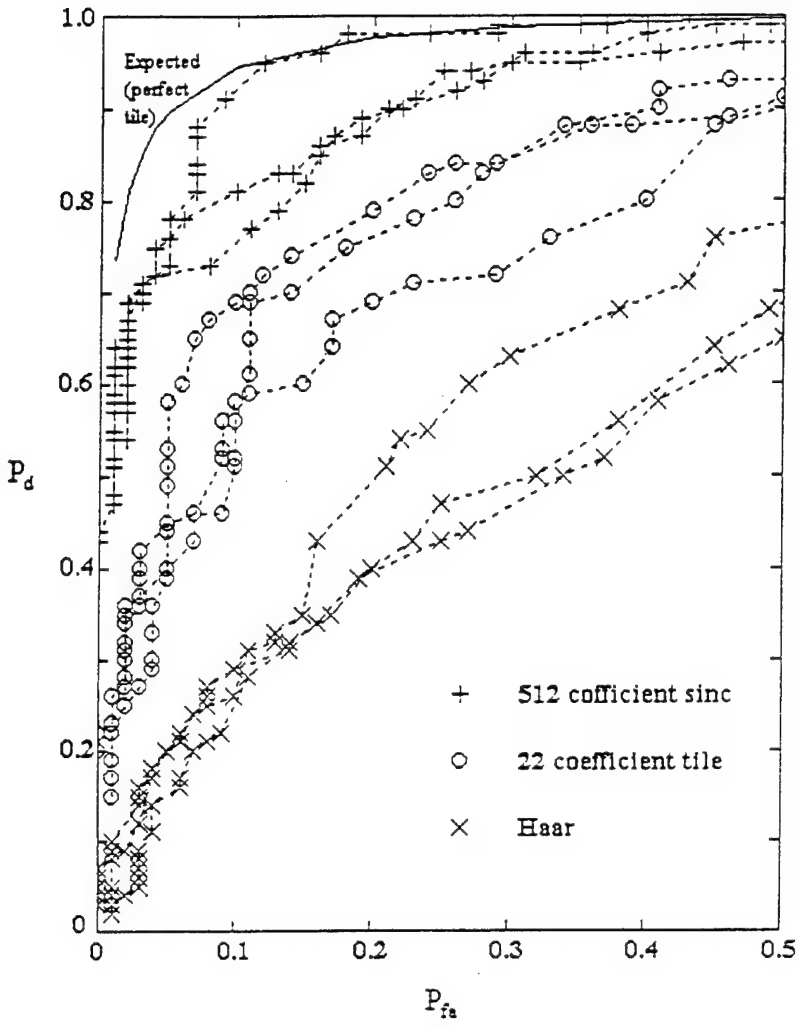



Figure 5.8. FBC Simulation Results

To figure out how much of a cell's energy would be covered, let's first consider a 1x3 block. In this case, if the cell is centered on the tiles in the frequency dimension, as we show in the upper part of Figure 5.10, we would have 0.467 (if the β layer tiles' frequency dimension is equal to B/2) to 0.774 (if the β layer tiles' frequency dimension is equal to B) of the cell's energy. On the other hand, if the cell were as off center as possible, to the point that an edge of the tiles started at the center of the cell's sinc squared energy distribution, the block would only contain 0.387 to 0.4515 of the cell's energy.

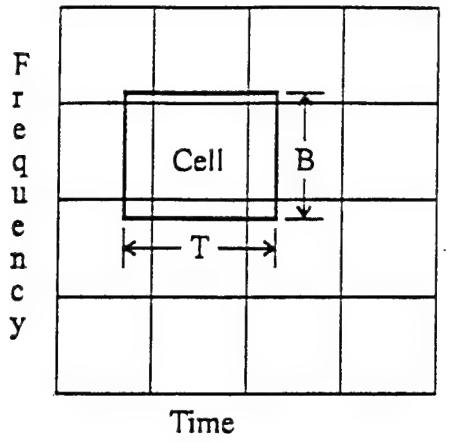



Figure 5.9. Tiling at the β Layer

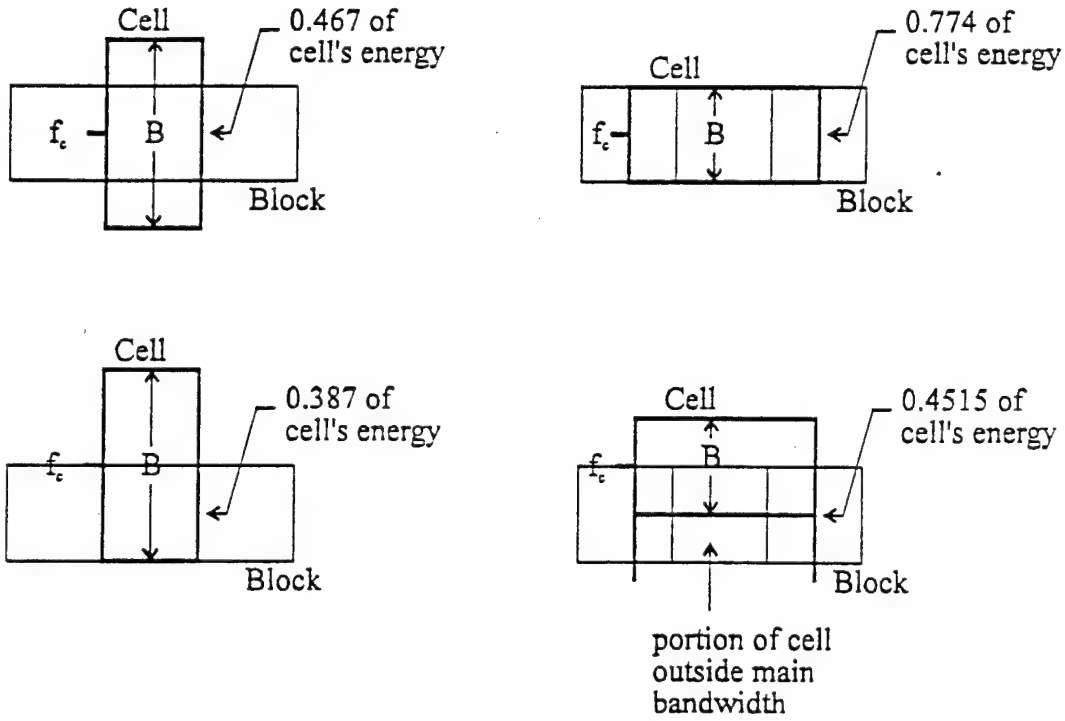



Figure 5.10. How a 1x3 Block Can Cover a Cell

With a 2x3 block the situation gets slightly more complicated. As we show in Figure 5.11, if the cell happens to be centered on the block, the block will have 0.774 to 0.903 of the cell's energy. To find the worst case, though, we must realize that blocks overlap in the frequency dimension. In other words, two blocks may have a row of tiles in common. In this case, we want to consider whichever of the two blocks contains the most cell energy. That will be the one whose center is

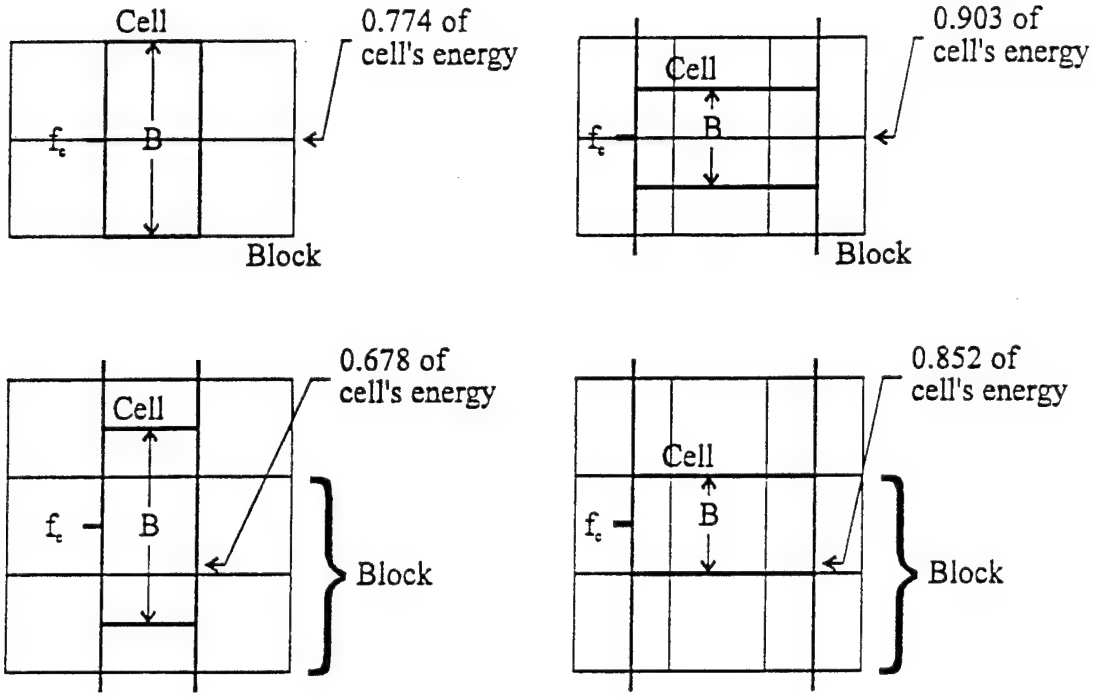



Figure 5.11. How a 2x3 Block Can Cover a Cell

closest to the main lobe of the cell. The extreme in this case, then, would be for the main lobe to be centered right in the middle of one of the rows of tiles. In this case, the block with most of the cell's energy would only have 0.678 to 0.852 of the total.

3x3 and 4x3 blocks can be evaluated in a similar manner. The greatest and least amounts of a cell's energy that different size blocks can have are shown in Table IV. As the discussion above indicates, there are two factors that go into determining the amount of cell energy: The cell's position relative to the tiles, and the cell's dimensions relative to the tiles. We will take the worst case, the least amount of cell energy that may be contained in a block, to continue our evaluation.

The next step is to consider the blocks containing only noise energy. In this case, we can compute the energy pdfs with (2.19) for 3, 6, 9, and 12 tiles giving us (with our usual normalized noise energy) Chi Squared curves with 3, 6, 9, and 12 degrees of freedom. For a given threshold, the probability of false alarm, $P_{fa}$, for each of these curves can be computed by integrating from the threshold value to infinity. The threshold values to give us a $P_{fa}$ of 0.1 (picked for convenience) are shown in Table IV.

**Table IV**

Amount of Cell Energy Collected With Different Size Blocks at the β Layer and $P_d$ When the Threshold is Set For $P_{fa} = 0.1$

| Block Size freq x time | Minimum Energy | Maximum Energy | Energy Difference | Th for a $P_{fa}$ of 0.1 | $P_d$ with $\varepsilon_c = 10$ |
|---|---|---|---|---|---|
| 1 x 3 | 0.387 | 0.774 | 0.387 | 6.251 | 0.473 |
| 2 x 3 | 0.678 | 0.903 | 0.225 | 10.645 | 0.583 |
| 3 x 3 | 0.838 | 0.931 | 0.093 | 14.684 | 0.605 |
| 4 x 3 | 0.899 | 0.950 | 0.051 | 18.549 | 0.584 |

We now can compute the energy pdfs for cell plus noise energy with (2.29), giving us Chi-Squared curves with 3, 6, 9, and 12 degrees of freedom and non-centrality parameters equal to the values for least amount of cell energy per block. Then, using the threshold values described above, the probability of detection, $P_d$, can be calculated. These values are also shown in Table IV, when we have a total cell energy of 10. Other cell energies give proportionally similar results. The point of this is that the best results (highest $P_d$) are for a 3x3 block, hence the "nine tile scheme" appears to be best for detection.

To show the β layer is the best QMF layer to use for detection, Tables V and VI show results for the "β - 1 layer," the layer of the QMF bank tree before the β layer where the tiles are twice as tall in the frequency dimension and half as long in time, and the "β + 1 layer," the layer of the QMF bank tree after the β layer where the tiles are half as tall in the frequency dimension and twice as long in time. Just as the block length in time for the β layer should be 3 tiles, the length for the β - 1 layer should be 5 tiles to cover the cell, and the length of the β + 1 layer should be 2 tiles. As the tables show, none of the results for $P_d$ exceed those of the 3x3 block in Table IV.

**Table V**

Amount of Cell Energy Collected With Different Size Blocks at the β - 1 Layer and $P_d$ When the Threshold is Set For $P_{fa} = 0.1$

| Block Size freq x time | Minimum Energy | Maximum Energy | Energy Difference | Th for a $P_{fa}$ of 0.1 | $P_d$ with $\varepsilon_c = 10$ |
|---|---|---|---|---|---|
| 1 x 5 | 0.452 | 0.903 | 0.451 | 9.236 | 0.454 |
| 2 x 5 | 0.852 | 0.950 | 0.098 | 15.987 | 0.593 |
| 3 x 5 | 0.927 | 0.966 | 0.039 | 22.307 | 0.556 |
| 4 x 5 | 0.945 | 0.975 | 0.030 | 28.412 | 0.510 |

Table VI

Amount of Cell Energy Collected With Different Size Blocks at the $\beta + 1$ Layer and $P_d$ When the Threshold is Set For $P_{fa} = 0.1$

| Block Size freq x time | Minimum Energy | Maximum Energy | Energy Difference | Th for a $P_{fa}$ of 0.1 | $P_d$ with $\varepsilon_c = 10$ |
|---|---|---|---|---|---|
| 3 x 2 | 0.621 | 0.889 | 0.268 | 10.645 | 0.546 |
| 4 x 2 | 0.751 | 0.903 | 0.152 | 13.362 | 0.577 |
| 5 x 2 | 0.832 | 0.909 | 0.077 | 15.987 | 0.583 |
| 6 x 2 | 0.876 | 0.931 | 0.055 | 18.549 | 0.572 |

Of course, one key point in the discussion above was that the cell must be contained within the block. The way we accomplish this is to first take the entire time frequency plane of interest and compute the energy in every set of 3x3 adjacent tiles. The list of these blocks are then sorted, from greatest to least energy. The first block on the list is than taken and overlapping blocks are discarded. This process is continued down the list, and the result should be a list of blocks, some containing the amount of cell energy in the range indicated in Table IV plus noise energy, and the rest containing only noise energy and energy from outside the main bandwidth of the cell. In a receiver designed like the diagram in Figure 1.2, this list of blocks would be sent from the analyzer to the classifier to determine whether a signal or signals are present. The Matlab file to accomplish the nine tile scheme is listed and described in Appendix B.

### Analyzing the Nine Tile Scheme

To determine how well the nine tile scheme works for detection, we need to know the pdf curves for the energy in the blocks for the cases where the input consists of noise only, and where it consists of signal and noise. Unfortunately, as we will see, these curves cannot be determined theoretically.

As we saw above, before discarding overlapping blocks, the energy pdf curves were Chi-Squared with nine degrees of freedom (with or without a non-centrality parameter). In the nine tile scheme, however, we discard from zero to 24 overlapping blocks, all with less energy. Analysis of this situation comes under a branch of statistics called "order statistics," and books have been devoted the subject [2] [3] [5] [13]. In order statistics, a number of random variables (RVs) are drawn and arranged in either ascending or descending order. One goal is then to determine the pdfs for the RV in the first, second, etc. positions. Unfortunately, practically all of the useful development in order statistics relies on the statistical independence of the RVs being ordered, a case we do not have here, since the blocks have tiles in common.

For what it is worth, we can explore what goes into the pdfs we desire. We begin by taking the highest energy block and eliminating overlapping blocks. If this is the first time doing this, and the highest energy block is not near one of the edges of the time frequency region being examined, we will eliminate 24 overlapping blocks. If, however, there is a possibility overlapping blocks have been

eliminated previously, and/or some of the overlapping blocks don't exist because we are near the edge of our region of observation, fewer blocks will be eliminated.

Let $f_{m:n}(x)$ denote the pdf for the m-th RV out of n order statistics ($m \le n$) arranged so the n-th RV contains the most energy. We are interested in $f_{n:n}(x)$ for n from one to 25, where n represents the number of overlapping blocks discarded, plus the one saved. Now, as Figure 5.12 shows, there are several ways blocks can overlap. In the first case, one block overlaps at two tiles, while another overlaps at one of those two. In the second case, both overlapping blocks have one tile in common with the block saved. In the cases where it is possible for blocks to overlap in different ways, so the n overlapping blocks cover different amounts of the highest energy block, there will, in general, be different versions of $f_{n:n}(x)$. Without rigorously finding the number of possibilities for each value of n, we can denote each with (i), and each pdf as $f^{(i)}_{n:n}(x)$. Assuming a noise only input, we can also denote the probability of each of these possibilities occurring as $p^{(i)}_n$, and obtain the conditional pdf for the energy of the blocks

$$f_{9\ \text{tile}\ |\ \text{noise}}(x) = p_{25}\, f_{25:25}(x) + p_{24}\, f_{24:24}(x) + \sum_i p^{(i)}_{23}\, f^{(i)}_{23:23}(x) + \cdots + \sum_i p^{(i)}_{2}\, f^{(i)}_{2:2}(x) + p_1\, f_{1:1}(x) \tag{5.17}$$

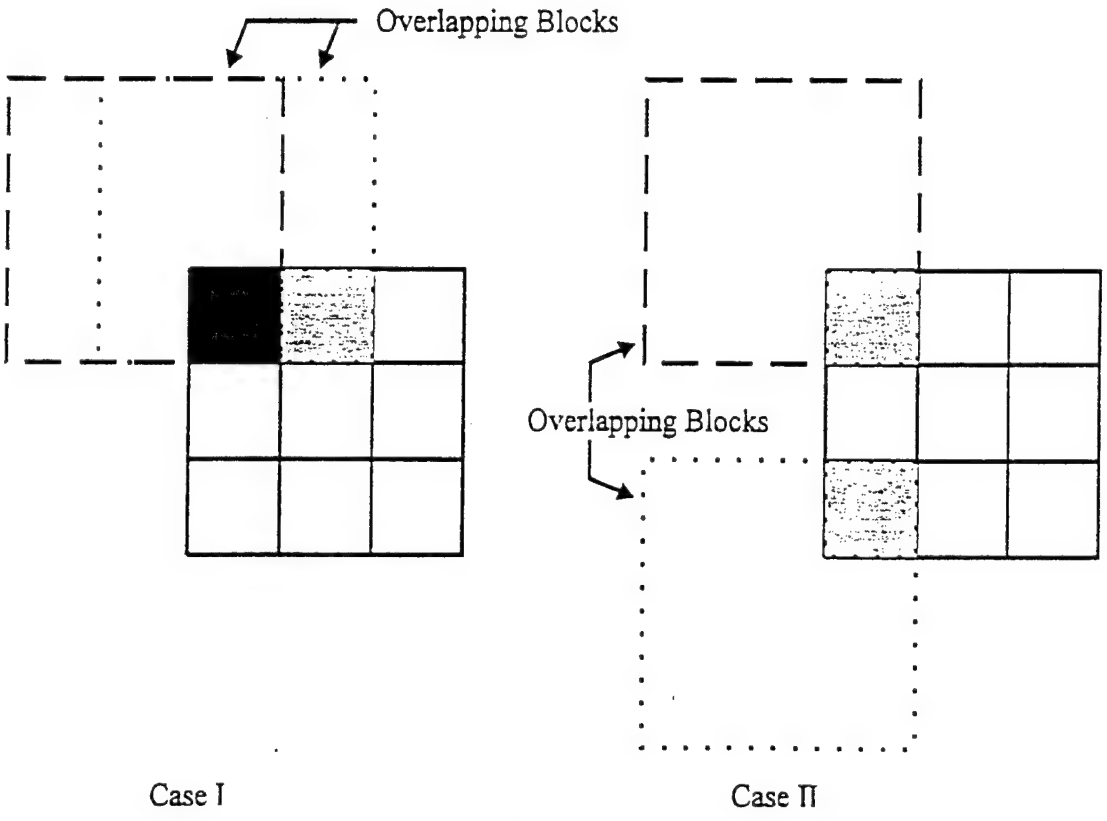



Figure 5.12. Two Ways Blocks Can Overlap

where we do not know any of the functions on the right hand side. A similar equation can be found for the case where signal plus noise are present, although the amount of signal energy in common between overlapping blocks would also have to be taken into account.

With all of these mathematical difficulties, evidently the only way to obtain these pdfs is empirically. The curve for the noise only case, in particular, can be obtained this way. An empirical pdf curve, created by passing 100 noise waveforms (with the variance for each equal to one) through the QMF bank tree with the energy concentration filter described in Chapter III, employing the nine tile scheme on the layer six output, binning the resulting tile energies and averaging the results for each bin, is shown in Figure 5.13. (The script file generating this curve is shown in Appendix B.) We will not use this data directly. Rather, we will only use the maximum energy block from each set of noise samples.

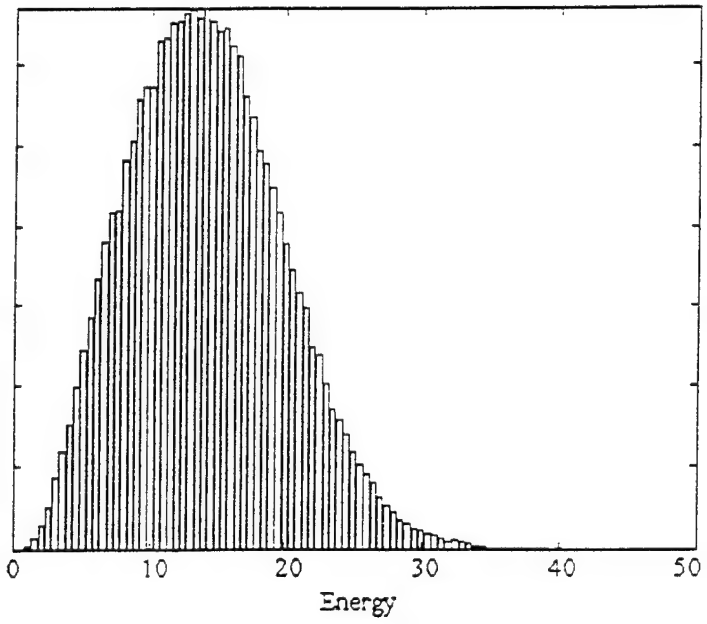



Figure 5.13. Empirical Energy Distribution For Nine Tile Scheme Output With a Noise Only Input

Even though there appears to be no way to analyze the scheme theoretically, we can bound our detection capabilities. As a lower bound, we can use a radiometer set to cover the portion of the time frequency plane containing the signal to be detected. In this case, the theoretical ROC curve can be computed by integrating under the Chi-Squared curve with the degrees of freedom equal to two times the time bandwidth product, and under the non-central Chi-Squared curve with the same degrees of freedom and with a non-centrality parameter equal to the signal energy in the observation period. We should note, however, the nine tile scheme may not work as well as the radiometer in all circumstances. This is a bound only in the sense that, if the nine tile scheme does not work as well, it makes more sense for the interceptor to use the radiometer, as it is less complex. Just as we found for

the FBC, we can predict that the nine tile scheme should have an advantage in cases where the signal(s) consist of relatively few cells in the observation period, with each cell containing a moderate amount of energy.

An upper bound for the nine tile scheme is easy to obtain. If we did know the channelization, cell dimensions, and hop synchronization of the signal(s) we were trying to detect, we could match the tiling to the cells, just as we did for the FBC. It is obvious that the nine tile scheme can not be better than this, and so a theoretical ROC curve obtained for an intercept receiver when these characteristics are assumed to be known must bound the nine tile curve.

**Analysis Results**

Just as with the outputs from the radiometer bank in the FBC, there are several ways to analyze the output of the nine tile scheme. Here we will consider one of the simplest: We will find the largest energy block at the $\beta$ layer, and compare that against a threshold. (Of course, in the receiver described in Figure 1.2, the entire list of blocks, their locations, and energies would be sent to the classifier block, and a detection scheme would be implemented there. We do not wish to explore the functions of the classifier block in any great detail in this report. However, for the sake of our simulations we must decide on a particular method for evaluating the data.) This particular method has the advantage of having some similarity to the method we used earlier with the FBC. In fact, as we will see shortly, the resulting ROC curves are remarkably similar.

The simulations were carried out using Matlab code listed and described in Appendix B. The input signals were similar to those used when simulating the FBC except here the cells were randomly offset from the tiles in both time and frequency. In these simulations, in addition to the nine tile scheme, the energy in all of the tiles in the observed bandwidth output from layer six of the filter bank, were collected, added, and compared to a threshold. This yielded simulated results for a radiometer with a bandwidth matching the signal bandwidth, over the observation time.

The ROC curves resulting from these simulations are shown in Figure 5.14. The signal parameters used are identical to those used for the FBC simulations (including the cell energy of $\varepsilon_c = 20$), except for the phase and channel shifts. Because simulations involving the 512 coefficient modified sinc filter were so time consuming, a 32 coefficient modified sinc filter was used instead. Because the Haar filters yielded poor performance in the FBC simulations, they were not used here. Once again, the modified sinc filter yields the better results.

In Figure 5.14, the theoretical radiometer ROC curve is also shown (calculated using Gaussian approximations to the Chi Squared pdf curves, as described in [41]). This figure shows our signal is such that both of the filters used in our QMF bank tree yield better results. As a verification that both the theory and simulation results are in agreement, Figure 5.15 compares the radiometer results obtained for the six sets of runs with both filters against the theoretical curve. As can be seen, the agreement is good.

Finally, Figures 5.16 and 5.17 compare the results of the FBC simulations to those described here for each of the two filters. (Note we are comparing a 512 coefficient modified sinc filter used for the FBC simulations against a 32 coefficient filter used for the nine tile scheme simulations.) Interestingly enough, there appears to be little difference. Evidently, then, at least within the bounds of our simulations, the main penalty for ignorance of the signal parameters to be detected is the increased computational cost of the nine tile scheme over the FBC.

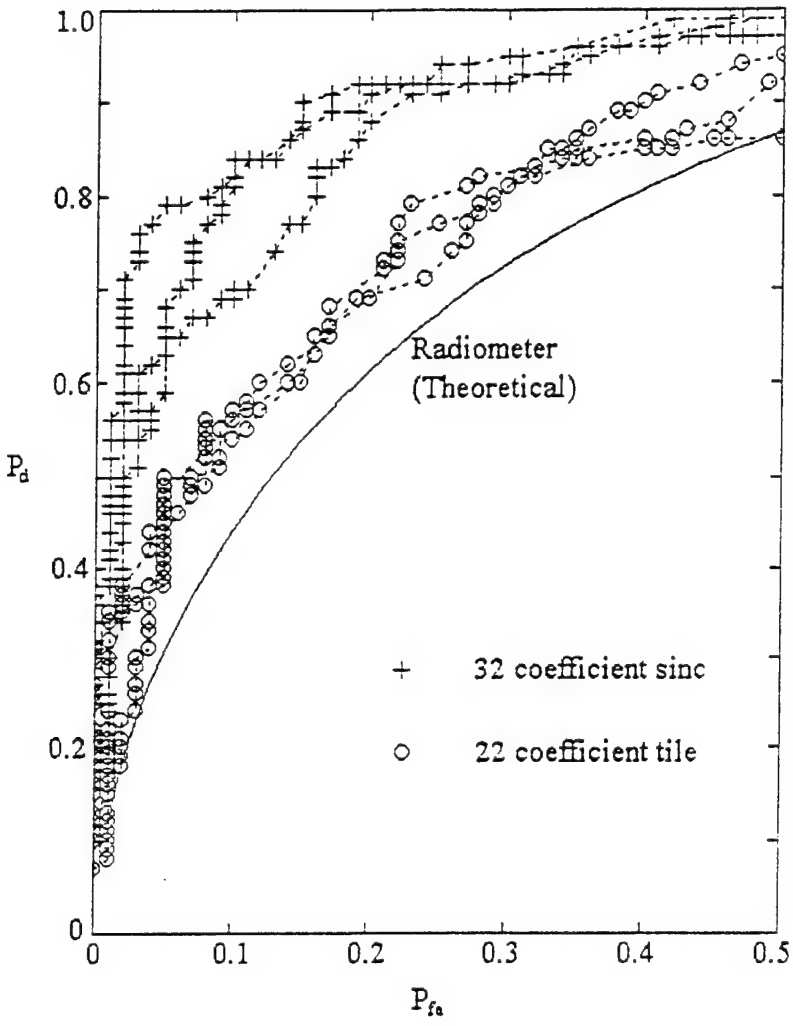



Figure 5.14. Nine Tile Scheme Analysis Results

**The Nine Tile Scheme With Unknown Hop Rate**

Finally, we consider the case where the interceptor does not know the received energy, channelization, synchronization, or hop rate of the Fast FH, TH, or FH/TH signals he is trying to detect. The most obvious approach in this case, given the discussion above, is to use the nine tile scheme on all of the layers of the QMF bank that may be a cell's β layer (where the tile length may be between 0.5 and 1.0 times the cell length in time). Then, the detector can perform some sort of test for detection on each layer, and combine the results for the various layers. In what follows, we will compare the results for this scenario against the case where the β layer is known.

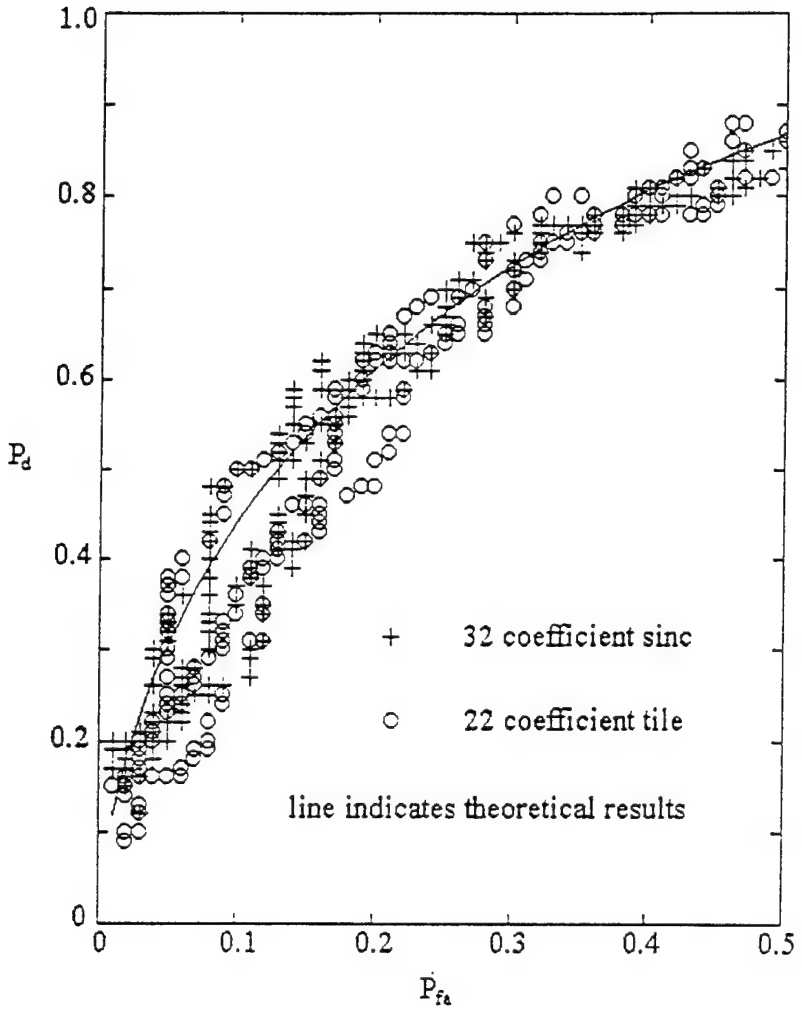



Figure 5.15. Comparison of Theoretical Radiometer Results With Observed Results

Let's consider a detection scheme, similar to the one we used previously, where only the largest energy block in each layer is considered, and a detection decision, for each layer, is made by comparing the energy in these blocks against a threshold. In the case where a single signal is present, if we only look at the β layer and detect the signal, looking at additional layers does not add additional information. If, on the other hand, we do not detect the signal in the β layer, our chances of detecting it in other layers are still less; although it is, of course, possible that a fortuitous signal and noise energy distribution would cause what, in effect, is a false alarm at another layer. It appears then, a logical way of combining the results from multiple layers is to use OR logic, and take a detection for any layer to be a detection for the system.

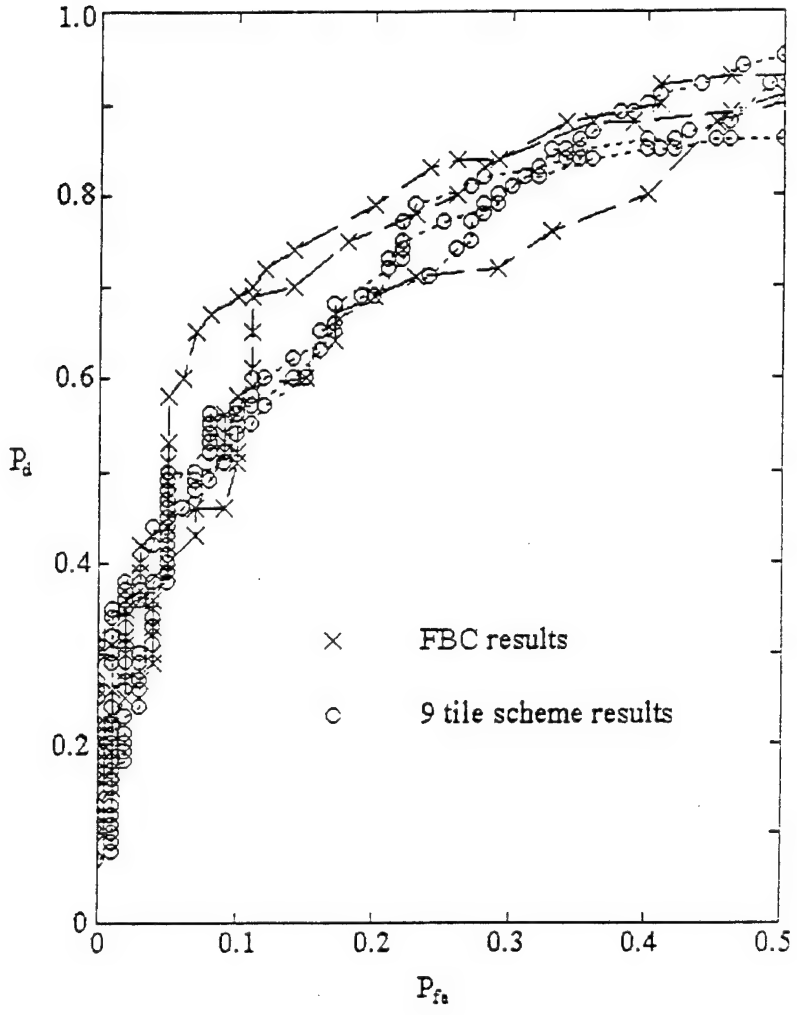



Figure 5.16. Comparison of Filter Bank Combiner Results and Nine Tile Scheme Results With 22 Coefficient Tile Filter

With this logic, the probability of detection increases only slightly from the case where the hop rate is known, so the overall value for $P_d$, relative to a given threshold, should be about the same. The probability of false alarm, on the other hand, when there are no signals present, will be the same for each layer, and for each layer will be the value for $P_{fa}$, relative to a given threshold, found for the case where the hop rate is known. If the data from the layers were statistically independent of each other, we could then calculate the overall probability of false alarm, $\Pi_{fa}$, in our present case to be

(5.18) $$\Pi_{fa} = 1-(1-P_{fa})^{NL}$$

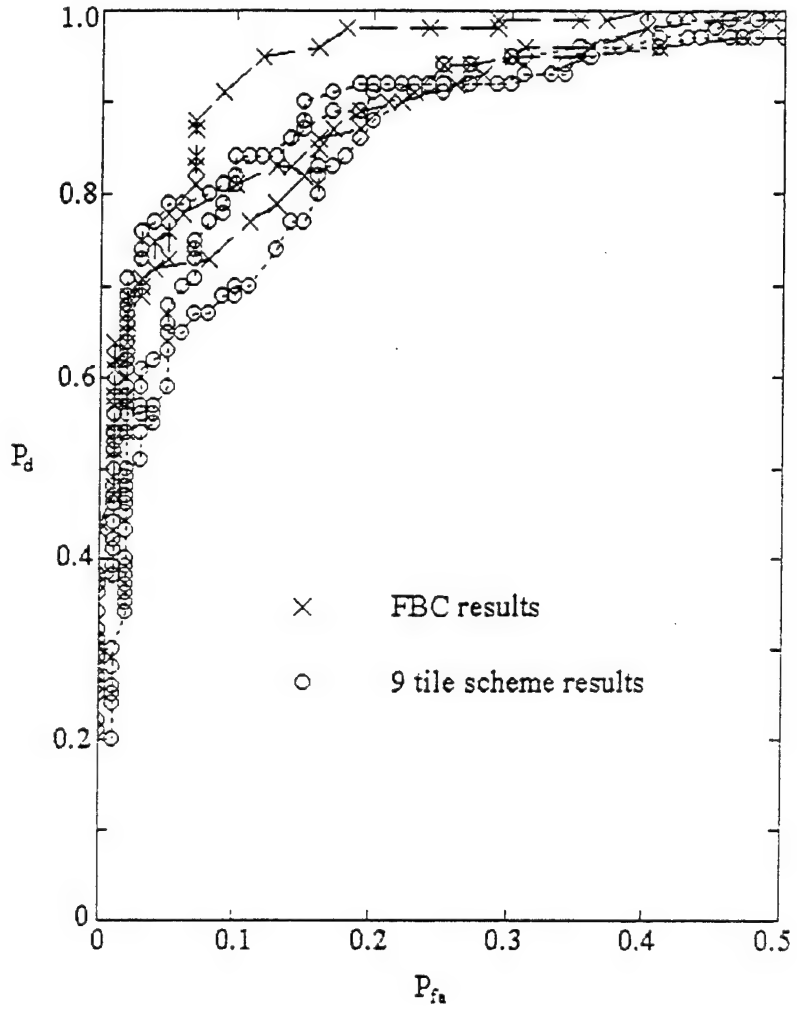



Figure 5.17. Comparison of Filter Bank Combiner Results and Nine Tile Scheme Results With 32 Coefficient Modified Sinc Filter

where NL are the number of layers examined. The layers are not independent, however, and so (5.18) should be regarded as a worst case upper bound on the probability of false alarm. Qualitatively then, we can say our probability of detection will not change very much from the case where we know the hop rate, but the probability of false alarm will increase up to the limit in (5.18). The ROC curve, then should be expected to shift to the right, yielding poorer results.

The Matlab code for this simulation analysis is listed in Appendix B. In these, layers three through ten were examined, the detection decision being made by comparing the largest energy block found among all of the layers against a threshold. Because of the length of time involved in running

these simulations, only 50 samples of signal and noise were examined for each set. The signals were identical to the ones used above in the case where the β layer was assumed to be known.

Results, for the 32 coefficient modified sinc filter, are shown in Figure 5.18. Also shown, for comparison, are the FBC results from Figure 5.8 and results modified by taking the nine tile scheme results shown in Figure 5.17, and finding $\Pi_{fa}$ as indicated by (5.18) with NL = 8. Of course, since the variance of the curves of Figure 5.17 is high for low values of $P_{fa}$, we should expect the variance of $\Pi_{fa}$ to be high for somewhat greater values. Never the less, we see the predicted worst case results are, indeed, worse than those actually observed.

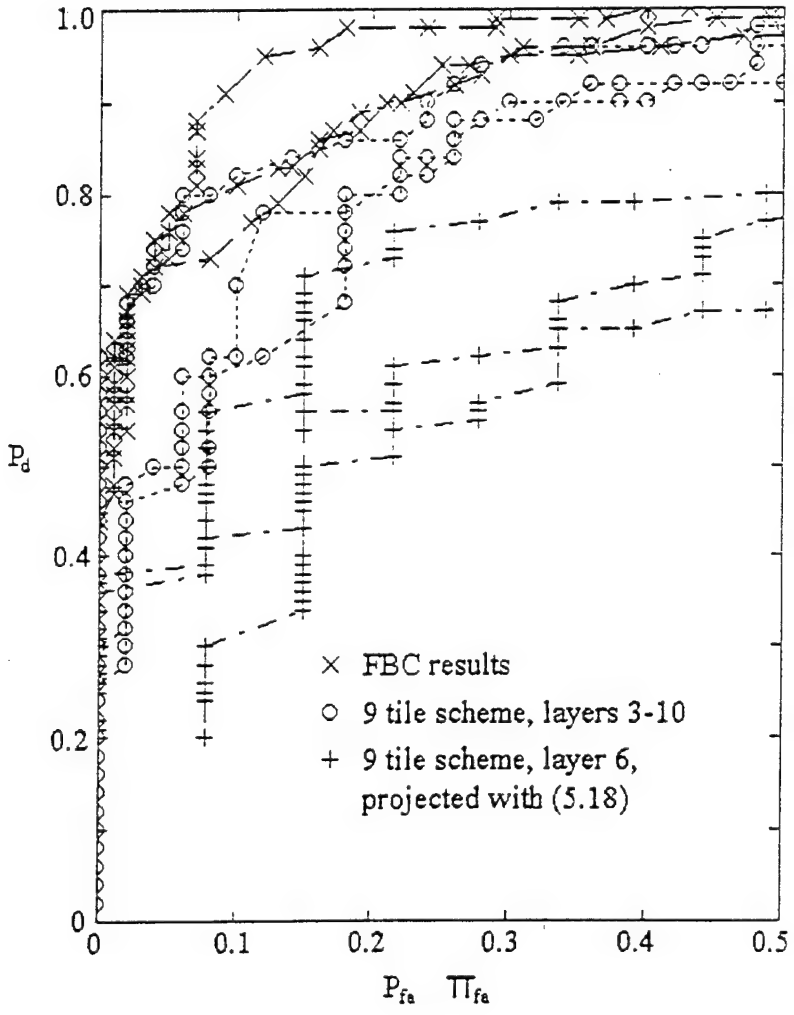



Figure 5.18. Nine Tile Scheme With Unknown Hop Rate.
Layers Three to Ten Examined With 32 Coefficient Modified Sinc Filter

This concludes our discussion of detection for the signals in this chapter. As we saw, the nine tile scheme, although difficult to analyze analytically, appears, at least in the simulations described here, to yield generally good results. Despite our assumptions that the interceptor does not know the channelization, timing synchronization, or hop rate of the signal to be intercepted, the performance is only slightly worse than the FBC, and the main cost to the interceptor is increased computation.

## Feature Extraction

We now consider the issue of feature extraction. Specifically, we are interested in features of individual cells such as the cell energy, cell dimensions, and the cells' location in the time frequency plane. In this section we will assume the interceptor receives enough cell energy that detection is not a problem.

### Strategy

Our strategy will be to use the nine tile scheme to obtain lists of high energy blocks for each layer that may be the $\beta$ layer for the cells of interest to the interceptor. From these lists, we then look for the highest energy block and save it, while discarding blocks from other layers that overlap in the time frequency plane. Our assumption in doing this is that the highest energy block (probably from a cell's $\beta$ layer) contains more of the cell's energy than the blocks from other layers. This procedure is repeated until all of the blocks in the original lists have been saved or discarded.

The result should be a list of blocks representing the cells. The blocks' energy should provide an estimate of the cells' energy, the blocks' positions in the time frequency plane should provide an estimate of the cells' positions, and the layers the blocks came from should provide an estimate of the cells' dimensions.

Earlier, 3 by 3 blocks--the nine tile scheme--were shown to be best for detection. The reasons for using the same scheme for feature extraction are not as analytically precise, and are worth some discussion. Because we will use the block energy to estimate cell energy, we would like as little variation as possible in the amount of cell energy picked up in the blocks. As Tables IV - VI show, the 3 by 3 blocks are good for this, but not the best. In fact, the variation decreases as the number of tiles in the block are increased. On the other hand, as the number of tiles are increased, the variation in noise energy increases. More importantly, however, in this case, is that the larger the blocks, the more likely a block is to contain more than a single cell. The 3 by 3 blocks seem like a good compromise, particularly since we are already using this size for detection.

Given the 3 by 3 tile blocks, let's examine what happens with blocks from different layers covering a particular cell. For layers less than the $\beta$ layer, the blocks will generally not cover the cell in the time dimension and, therefore, the $\beta$ layer block will contain more energy. Of course, as discussed above, the distribution of cell energy in the time dimension is not entirely deterministic, since it is partly a function of the cell carrier phase and interceptor's sampling rate, so it is possible that some of a particular cell's energy may be lumped at one end of a cell or the other. Also, there is a possibility that a cell in the $\beta$ - 1 layer that happens to have a length 2 to 3 times the tile length in time may be entirely contained within a block. That depends on the cell's position in the time frequency plane relative to the tiling grid. In these cases, since the blocks in layers less than the $\beta$ layer are taller in the frequency dimension and therefore pick up energy from the cell's sidelobes,

there is a chance they will contain more cell energy than the β layer. For layers greater than the β layer, the blocks will be shorter in the frequency dimension than the block in the β layer, and will again tend to contain less cell energy.

Of course, all of this discussion supposes ideal tiles. The non-flat passband and sidelobes in real tiles will cause further problems.

**Analysis Results**

The purpose of this simulation analysis was to determine how well the block positions and energies estimate the true cell positions and energies. In these, 25 runs per set were made. The signal parameters are similar to the ones used for detection, with 31 channels, 25 slots, 128 second long cells, and with the cells randomly offset from the tiles. One major difference between these runs and the ones described above for detection is that a signal cell energy of 256 was used here.

Layers five through seven were examined, and a list of blocks found using the procedure described above. Since at least ten cells from each signal run should occur during the observation interval, the ten highest energy blocks were collected and saved, for a total of 250 blocks from the 25 runs in each set. The block center positions were then compared to the known cell center positions. The energy for each of these blocks was also recorded. The results for these runs are shown in Figures 5.19 to 5.26. One set each was run for the 32 coefficient modified sinc filter and the 22 coefficient energy concentration filter.

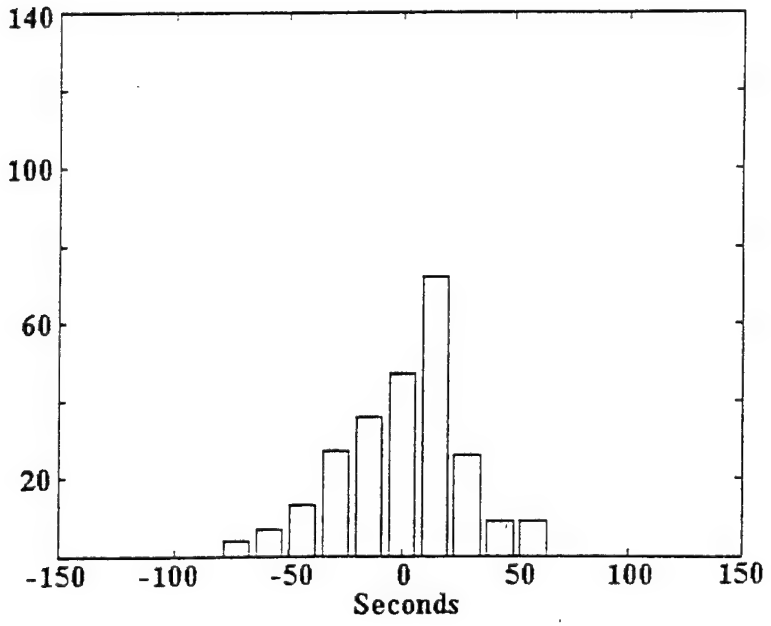



Figure 5.19. Error in Time Estimate With 32 Coefficient Modified Sinc Filter

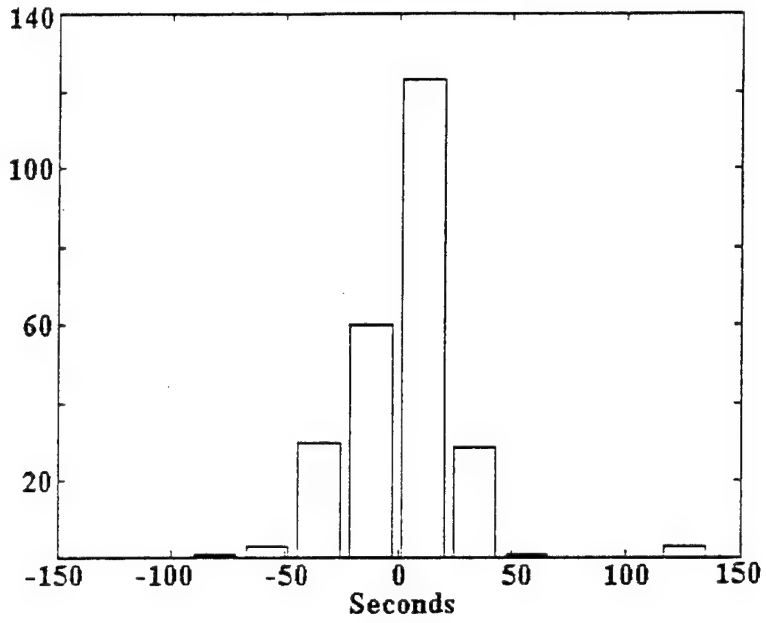



Figure 5.20. Error in Time Estimate With 22 Coefficient Energy Concentration Filter

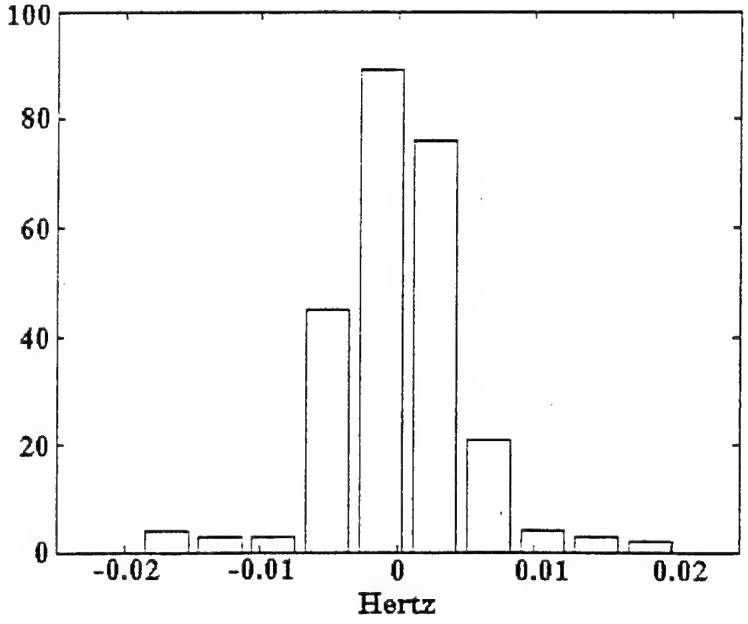



Figure 5.21. Error in Frequency Estimate With 32 Coefficient Modified Sinc Filter

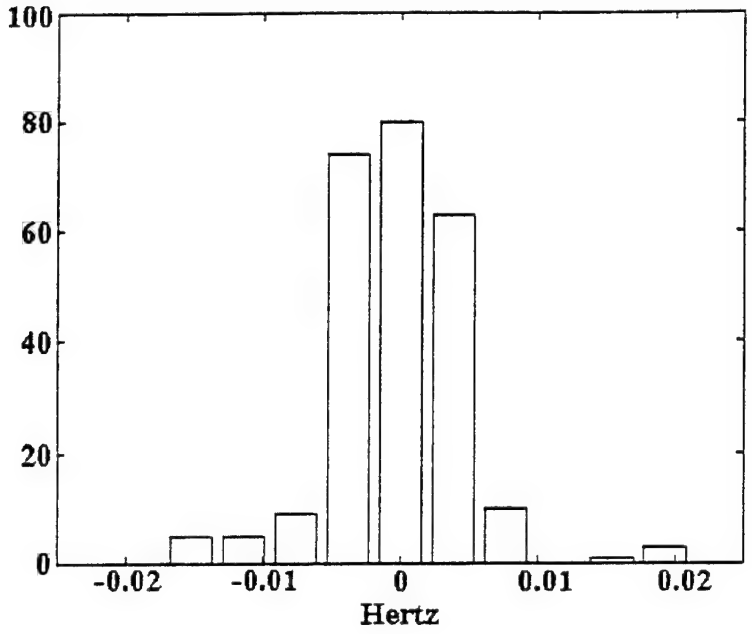



Figure 5.22. Error in Frequency Estimate With 22 Coefficient Energy Concentration Filter

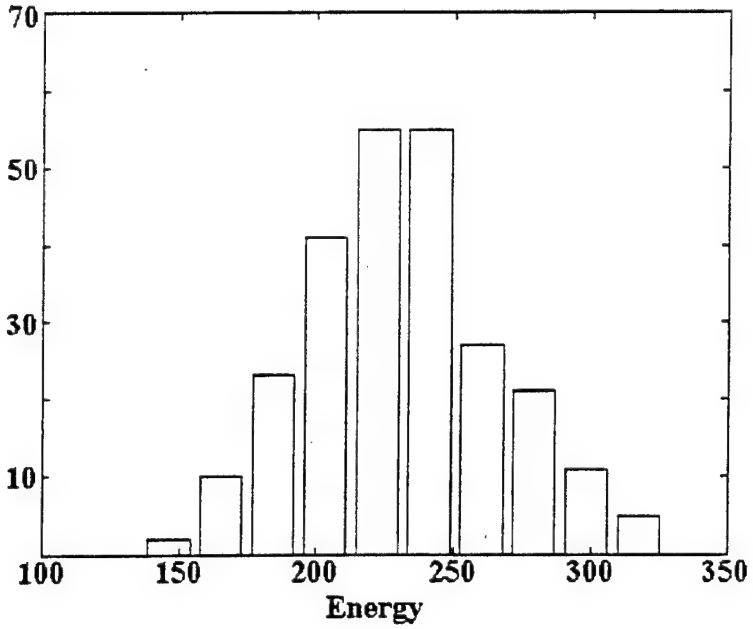



Figure 5.23. Block Energy Distribution (First 10 Blocks)
With 32 Coefficient Modified Sinc Filter

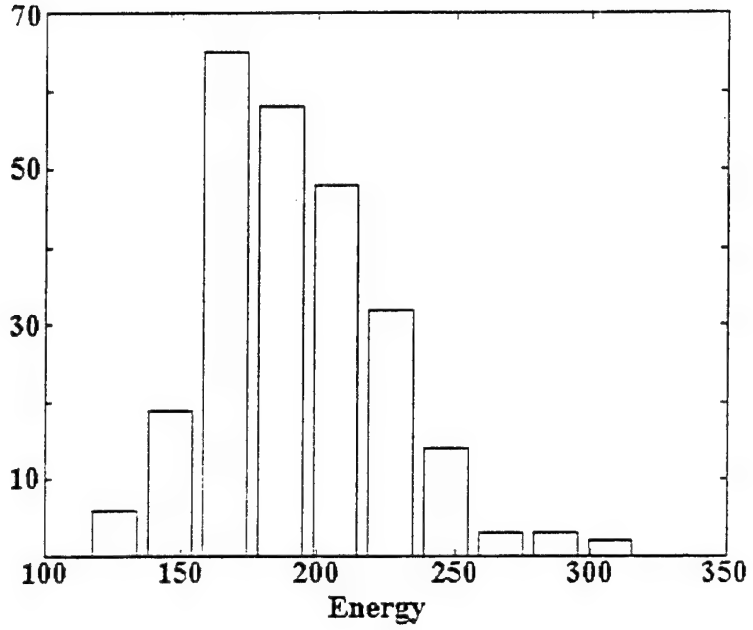



Figure 5.24. Block Energy Distribution (First 10 Blocks)
With 22 Coefficient Energy Concentration Filter

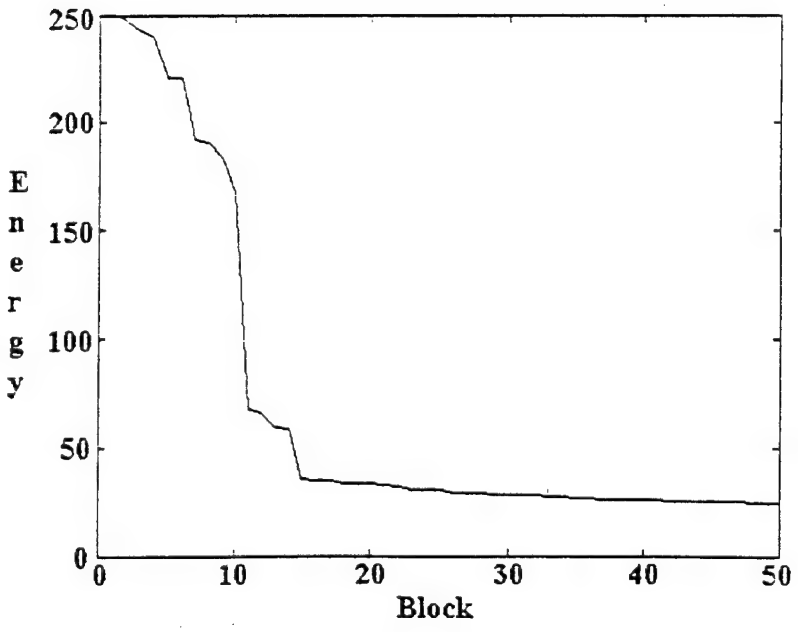



Figure 5.25. Energy in Highest 50 Blocks Found With
32 Coefficient Modified Sinc Filter

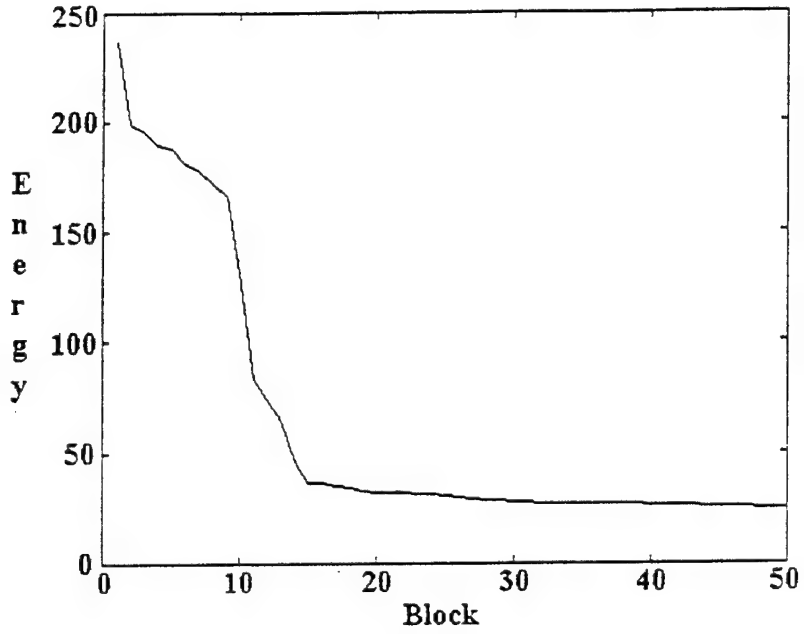



Figure 5.26. Energy in Highest 50 Blocks Found With
22 Coefficient Energy Concentration Filter

Histograms showing the distribution of time estimate errors are shown in Figures 5.19 and 5.20. The mean error for the 32 coefficient sinc filter is 1.596 seconds, and for the 22 coefficient energy concentration filter is 2.364 seconds. Histograms showing the distribution of frequency estimate errors are shown in Figures 5.21 and 5.22. The mean error for the 32 coefficient sinc filter is -0.00012 Hz, and for the 22 coefficient energy concentration filter is -0.00018 Hz. To consider the relevance of these values, it should be noted the $\beta$ layer for these signals is layer six, and the tile dimensions are 64 seconds by 0.0078125 Hz.

Histograms showing the distributions of energy in the first ten blocks from each run are shown in Figure 5.23 for the 32 coefficient modified sinc filter and in Figure 5.24 for the 22 coefficient energy concentration filter. For the modified sinc filter the mean energy is 230.5 with a standard deviation of 35.2. For the energy concentration filter, the mean is 192.1 with a standard deviation of 32.8.

Figures 5.25 and 5.26 show results of the last run of each set for the 32 coefficient modified sinc filter and 22 coefficient energy concentration filter. In these figures, the energies of the first 50 blocks, from the complete list (sorted by energy level) are shown. It is very obvious that the first ten blocks are those containing signal cells. This fact would be important in a receiver like the one shown in Figure 1.2, in which a classifier is used to separate blocks containing signals from those with noise only.

To examine the possibilities of using the maximum energy block to estimate the $\beta$ layer, single runs were made for FH/TH signals with cell lengths varying from 68 to 188 seconds. The

number of channels was adjusted in each case to ensure the signal remained within 0.125 to 0.375 Hz, and the number of slots was adjusted, to ensure there were at least ten cells in the observation interval. The amplitude of the signal was also adjusted to maintain a constant cell energy of 256. Layers from four to eight were examined, and the layers in which the ten highest energy blocks occurred were recorded. The numbers are shown in Tables VII and VIII.

**Table VII**

Layers in Which Maximum Energy Blocks Occurred For
32 Coefficient Modified Sinc Filter

| Cell Size Seconds | A | Slots | Channels | Layers | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 4 | 5 | 6 | 7 | 8 |
| 68 | 2.74 | 45 | 16 | 1 | 7 | 2 | 0 | 0 |
| 88 | 2.41 | 35 | 21 | 0 | 5 | 5 | 0 | 0 |
| 108 | 2.18 | 30 | 26 | 0 | 1 | 8 | 1 | 0 |
| 128 | 2.00 | 25 | 31 | 0 | 0 | 6 | 4 | 0 |
| 148 | 1.86 | 22 | 36 | 0 | 1 | 3 | 6 | 0 |
| 168 | 1.75 | 18 | 41 | 0 | 0 | 5 | 5 | 0 |
| 188 | 1.65 | 17 | 46 | 0 | 0 | 5 | 4 | 1 |

**Table VIII**

Layers in Which Maximum Energy Blocks Occurred For
22 Coefficient Energy Concentration Filter

| Cell Size Seconds | A | Slots | Channels | Layers | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 4 | 5 | 6 | 7 | 8 |
| 68 | 2.74 | 45 | 16 | 1 | 9 | 0 | 0 | 0 |
| 88 | 2.41 | 35 | 21 | 0 | 8 | 2 | 0 | 0 |
| 108 | 2.18 | 30 | 26 | 0 | 4 | 5 | 1 | 0 |
| 128 | 2.00 | 25 | 31 | 0 | 2 | 8 | 0 | 0 |
| 148 | 1.86 | 22 | 36 | 0 | 2 | 7 | 1 | 0 |
| 168 | 1.75 | 18 | 41 | 0 | 0 | 10 | 0 | 0 |
| 188 | 1.65 | 17 | 46 | 0 | 0 | 7 | 2 | 1 |

For the cell lengths from 68 up to and including 128, the β layer, as it was defined earlier in the chapter, is layer six. For the longer length cells examined in the table, the β layer is layer seven. As the tables show, many of the blocks with maximum energy are not from the β layer, but rather from layers immediately above and below. The reason for this can be seen when we re-look at why the β layer was picked to be significant. It was selected because it was the layer in which the cell time width, T, and bandwidth (when defined as 1/T) is guaranteed to be contained within a block of 3 by 3 tiles. There are cases, however, where a cell's position, relative to the tiles', may allow a significant amount of cell energy may be contained in blocks from the other layers. The algorithm, as it is described here, appears to offer a classifier enough information to estimate the β layer within , from which the classifier can estimate a signal's hop rate.

One goal of finding the cell features is to distinguish between multiple signals. This, of course, is the job of the classifier and is beyond the scope of this research. However, to get an indication of how this might work, several runs were performed with two FH/TH signals.

In the first of these, two signals, with equal cell dimensions, but greatly varying energies, were used. The first signal was identical to the one used in Figures 5.25 and 5.26 with 31 channels, 25 slots, and a cell duration of 128 seconds, except a cell energy of 1024 was used. The second signal also had 31 channels, 25 slots, and a cell duration of 128 seconds. This signal had a cell energy of 256. Figures 5.27 and 5.28 show the results for the 32 coefficient modified sinc filter and the 22 coefficient energy concentration filter. With the modified sinc filter, it is easy to distinguish between the ten blocks containing the higher energy signal, the ten containing the lower energy signal, and the rest of the blocks. With the energy concentration filter, on the other hand, distinguishing between the blocks containing the lower energy signal and blocks not containing cells is not as easy. A close comparison of the block position and the true cell positions reveals the blocks from 11 to 20 do represent the lower energy cells. The blocks with lower energy are due to sidelobes and out of tile energy collected from the higher energy cells. Since these "false alarms" primarily show up in blocks with the same time dimensions but different frequency dimensions as the true high energy signal cells, an intelligent classifier would still have hopes of recognizing them for what they are, and discarding them.

In the second test involving two signals, cell energies were set equal, but the cell dimensions were set to be different. The first signal was the FH/TH signal used in Figures 5.25 and 5.26 (31 channels, 25 slots). This signal had a cell duration of 128, so its β layer is six. The second signal was a FH/TH signal with cell duration of 1024 seconds, cell energy of 256, 255 channels and 3 slots. This signal's β layer is nine. One run each was made for the 32 coefficient modified sinc filter and the 22 coefficient energy concentration filter. The results for the 20 highest energy blocks are shown in Table IX. As can be seen, the modified sinc filter appears to do slightly better at distinguishing between the signals.

## Summary

In this chapter, we began by looking in detail at the energy distribution in the time frequency plane of spread spectrum signals consisting of cells with time bandwidth products of one. We then looked at detection, and found that, unless we exploit our knowledge of the energy concentrations due to these cells, the best detector we can find is the radiometer, or energy detector, set to cover the entire time frequency plane of interest.

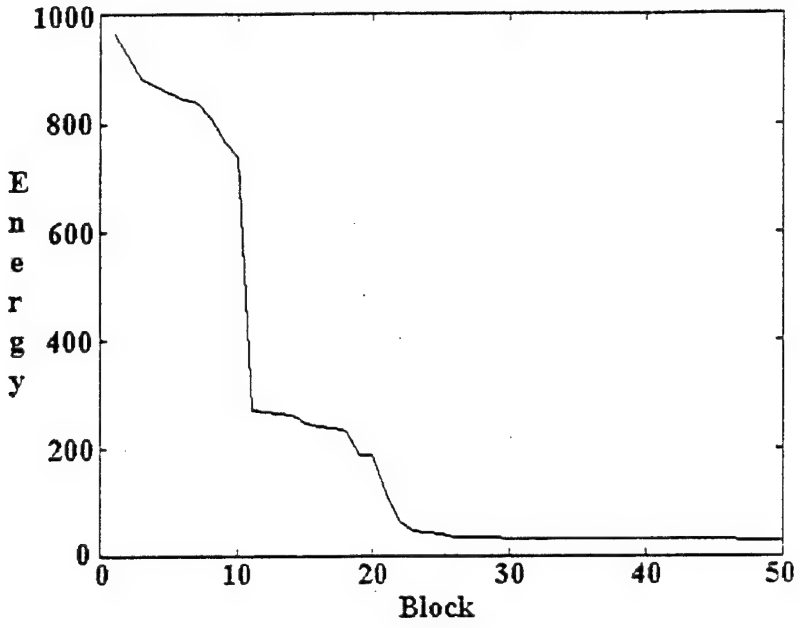



Figure 5.27. Energy in Highest 50 Blocks Found With 32 Coefficient Modified Sinc Filter When There Are Two FH/TH Signals Present

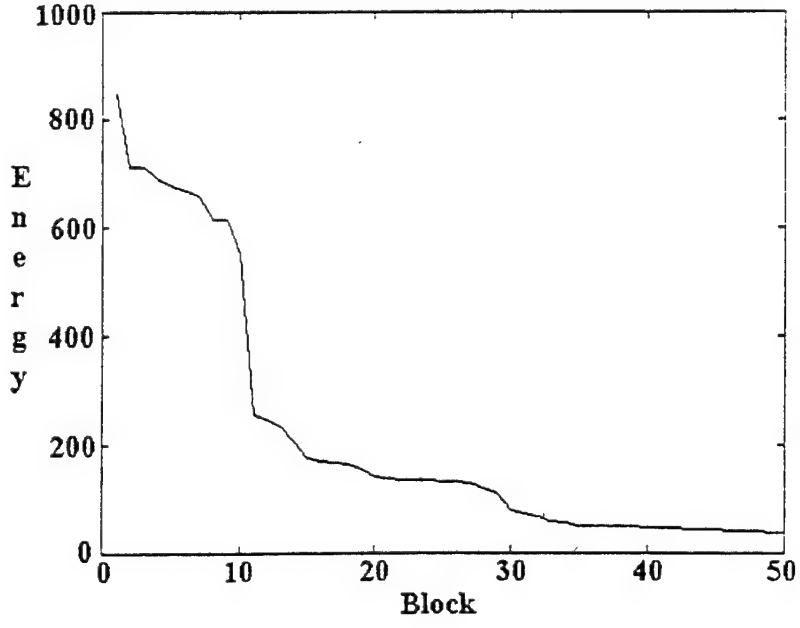



Figure 5.28. Energy in Highest 50 Blocks Found With 22 Coefficient Energy Concentration Filter When There Are Two FH/TH Signals Present

**Table IX**

Layers in Which Maximum Energy Blocks Occurred For
Two Signals With Different Hop Rates

| Filter | Layers | | | | | | |
|---|---|---|---|---|---|---|---|
| | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Modified Sinc | 0 | 8 | 1 | 4 | 5 | 2 | 0 |
| Energy Concentration | 0 | 6 | 2 | 7 | 3 | 2 | 0 |

We then took a different tack, and found that if an interceptor knows everything about a signal to be detected except for the received signal energy and the specific hop sequence, the best detector architecture is the filter bank combiner (FBC). This gave us a "best case" to measure our subsequent performance results against. Simulations were also run to verify an FBC implemented with the QMF bank tree performed as predicted.

Then we assumed our interceptor does not know the channelization, hop synchronization, or (later) cell dimensions of the spread spectrum signal, and developed the "nine tile scheme" to best exploit the output of the QMF bank tree for detection. The performance analysis for this scheme was shown to be mathematically intractable, and so simulation was resorted to--indicating a performance that is promising.

Finally we turned to feature extraction, and developed a strategy to use the nine tile scheme data to estimate the cells' energy, positions in the time frequency plane, and dimensions. Here, just enough data was collected to indicate that the scheme should work by providing a classifier enough information both to estimate overall signal parameters and to allow the classifier to distinguish between transmitters, at least in some cases.

Obviously, much more work could be done in the areas explored in this chapter. One key feature that could be added to the analysis would be a specific classifier. Then particular more-or-less complex detection algorithms could be explored. For example: Requiring a certain number of blocks, not overlapping in time, to exceed a threshold for a detection to be declared would be analogous to adding the binary moving window to the FBC. The same situation would apply to feature extraction, where a sophisticated classifier could work with the characteristics of the specific filters in the QMF bank tree to eliminate the effects of sidelobes, such as the ones described in association with Figure 5.28.

Of course, with a specific type of classifier, and a specific detection scenario, particularly where a range of "real world" signals are examined, the ultimate limitations of this receiver could be found. Specifically, the maximum number of signals that could be distinguished in a certain time frequency area, as well as the minimum separation of signal characteristics such as cell energy and dimension, could be determined.

## VI. Interception of Direct Sequence (DS) Signals

### Introduction

In this chapter we look at the detection and feature extraction of DS signals. Here, "detection" refers to the process, by the interceptor, of determining if one or more DS signals are present. For DS signals, "feature extraction" will refer to determining the signals' band edges and energy.

For our analysis, we will assume the DS signal, when present, is present for the entire observation interval, and that its energy is distributed across the frequency dimension with a sinc squared distribution. In this chapter we will not consider changes in energy distribution noted when looking at the signal over short time intervals. (We will assume the characteristics of these changes are unknown to the interceptor). For this reason, we will begin by taking the tiling from a given layer of our QMF bank tree and add the energies from tiles at specific frequencies across the observation interval, as shown in Figure 6.1.

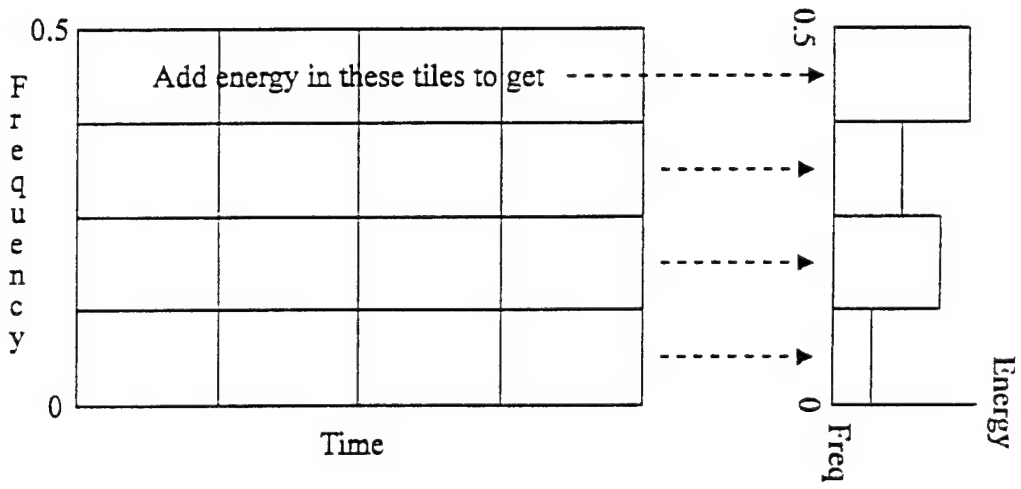



Figure 6.1. Adding Energy Across the Observation Time to Obtain a Spectral Vector

The result is a vector of values representing the waveform's energy across the spectrum--a "spectral vector" which is virtually identical to the results of a Fourier analysis with a suitable window. Because the computational cost of the QMF bank tree is higher than that of a similarly sized fast Fourier transform (FFT), the FFT is obviously a better solution if the only objectives of the intercept receiver are those of this chapter. If, on the other hand, the intercept receiver is also required to detect and distinguish between various types of hopped signals, and/or it is required to make an analysis of the internal energy distribution of the DS signal(s) (as discussed in Chapter VIII), the decomposition by the QMF bank tree will be needed anyway, and its results can be used for our purposes here.

Because we desire very sharp transitions in the tiles' frequency dimensions, and have little concern about the time characteristics, we will use the modified sinc filter for our analyses in this chapter.

We begin the chapter with a very quick discussion of the DS signal and its characteristics. We examine the detection of this signal when the location in frequency and bandwidth of the signal are known to the interceptor. Analytical and simulation results are then compared.

We then consider the case where the interceptor knows neither the center frequencies of the signal nor the signal's bandwidth, and we derive a suitable detection algorithm. Finally, we extend this algorithm to include feature extraction. (As we will see, this is almost automatic, and does not require any major modification to the algorithm.) Simulation results are then discussed and compared to analytical predictions.

### Direct Sequence Signals

A communications system using DS signals is shown in Figure 6.2. Here a carrier signal, which will generally already have information modulated on it, is further modulated with a wide band pseudo-noise signal. The term "pseudo-noise" as used here, refers to the fact that, although the signal look like noise to the casual observer, the intended receiver has an exact copy. There are several ways to carry this out in practice. For example, the pseudo-noise may be generated by some deterministic algorithm known only to the transmitter and receiver. On the other hand, a truly random signal may be recorded ahead of time, with copies of the signal sent to both the transmitter and receiver by some other means (compact disks sent by diplomatic pouch, for example). Regardless of the nature of the signal, the intended receiver somehow synchronizes its version with the transmitter's, and then demultiplexes the received signal, obtaining an estimate of the original.

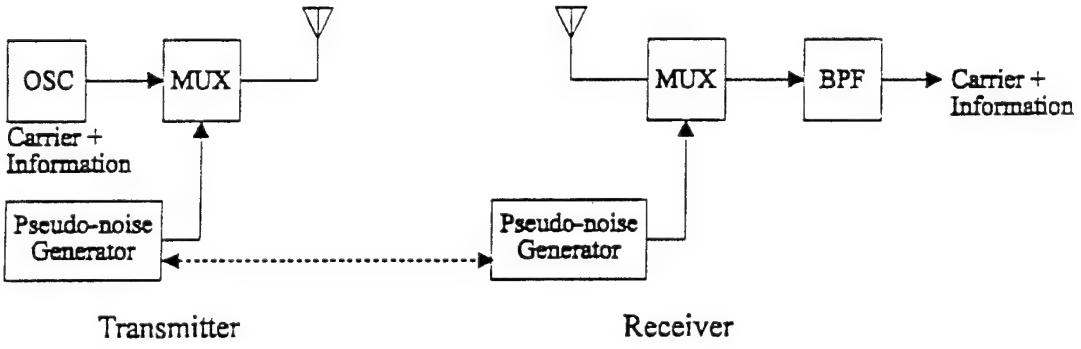



Figure 6.2. Direct Sequence Communications System

Often, the pseudo-random signal is a random binary waveform [17]. In a random binary waveform, the signal's value is randomly set to ±1 at fixed intervals. An example of this waveform is shown in Figure 6.3. The energy distribution in the frequency domain is sinc squared, with a main lobe null to null bandwidth of 2B, for B as defined in Figure 6.3. When the random binary waveform is used to modulate a sine wave carrier with frequency $f_c$, the resulting energy distribution is as

shown in Figure 6.4. This is the energy distribution of the DS signals we will consider in this chapter.

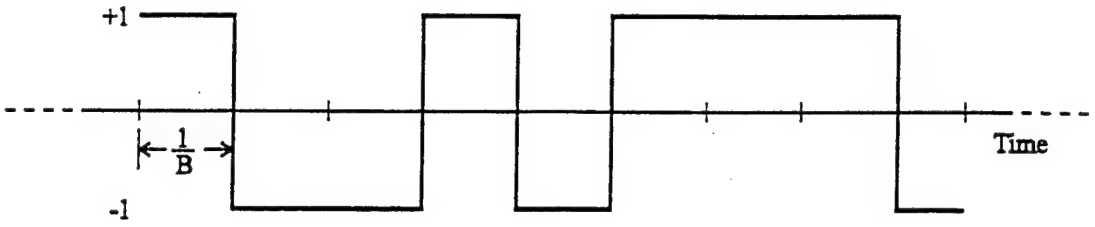



Figure 6.3. Example of Random Binary Waveform

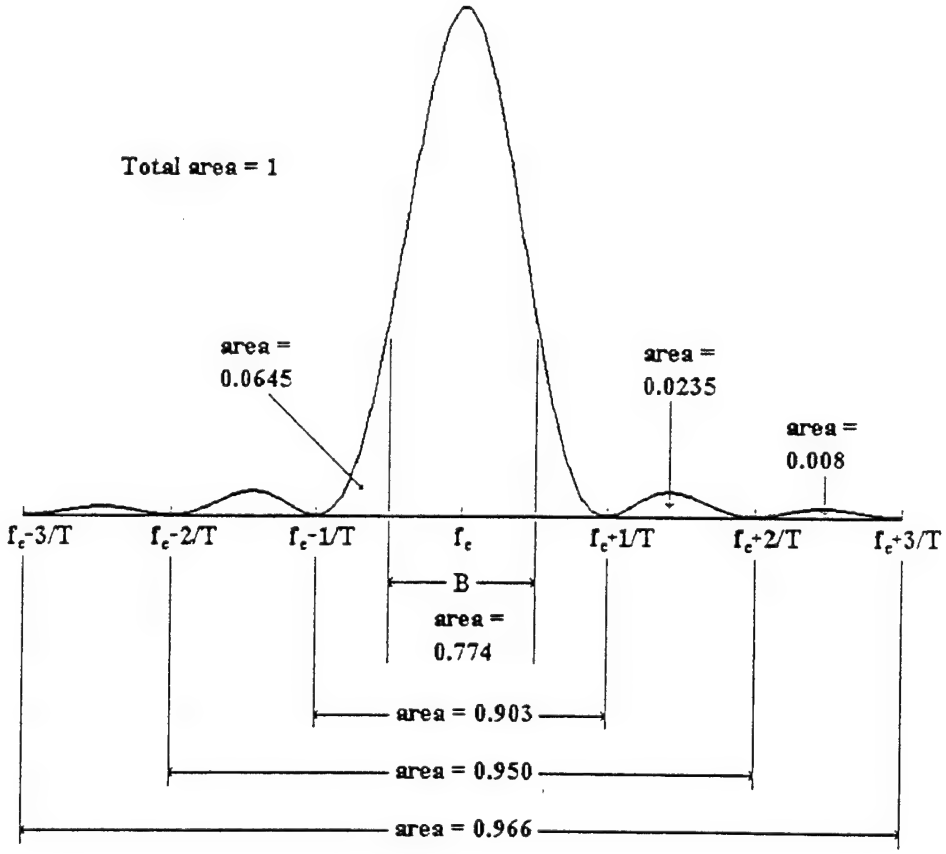



Figure 6.4. The Sinc Squared Function, With the Areas Under Key Portions of the Curve

The energy of the DS signal is

$$E = \frac{A^2 T}{2} \tag{6.1}$$

where T is the observation time, and A is the signal's peak amplitude.

**Detection**

We begin this section with a question: Let's say we have a DS signal with the energy distribution shown in Figure 6.4, and that this signal is embedded in white Gaussian noise (WGN). We wish to make a detection decision using a radiometer/threshold scheme like that shown in Figure 6.5, with an integration time set to our observation interval and with an ideal brick wall filter centered at $f_c$ with a bandwidth of W. What is the best value for W? In other words: What W will maximize the probability of detection, $P_d$, for a given probability of false alarm, $P_{fa}$?

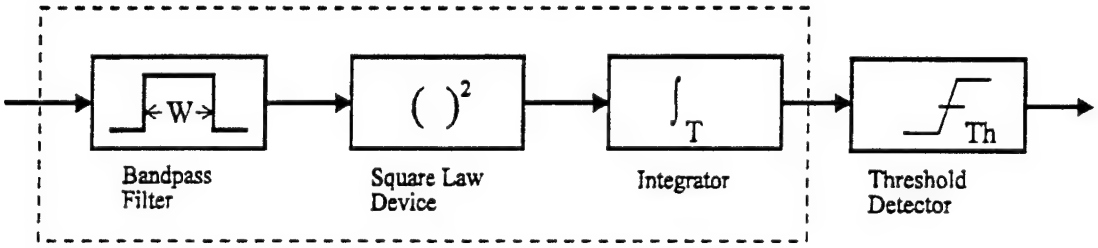



Figure 6.5. Radiometer With Threshold Detector

To answer this, we first note the signal energy collected by the radiometer will be

$$\varepsilon_c = \frac{E}{B}\int_{-W/2}^{W/2} \mathrm{sinc}^2(x/B)dx = \frac{2E}{B}\int_{0}^{W/2} \mathrm{sinc}^2(x/B)dx$$

$$= 2E\int_{0}^{W/2B} \mathrm{sinc}^2(y)\,dy \tag{6.2}$$

where E is the total signal energy from (6.1). Next, we know the pdf of the radiometer output will be Chi-Squared, with or without a non-centrality parameter depending on whether the signal is present. However, for large time bandwidth products, these curves can be approximated with Gaussian pdfs [41]. For a suitably large observation time this will be our case. With this approximation, $P_{fa}$ and $P_d$ will be

(6.3) $$P_{fa} = \tfrac{1}{2}\mathrm{erfc}\left[\frac{Th - N_0TW}{\sqrt{2N_0{}^2TW}}\right]$$

and

(6.4) $$P_d = \tfrac{1}{2}\mathrm{erfc}\left[\frac{Th - N_0TW - \varepsilon_c}{\sqrt{2N_0{}^2TW + 4N_0\varepsilon_c}}\right]$$

where

| | |
|---|---|
| Th | is the threshold at the radiometer output |
| $N_0$ | is the one sided noise density (we will usually normalize this so $N_0 = 2$) |
| T | is the observation time |

and erfc(x) is the "error function"

(6.5) $$\mathrm{erfc}(x) \equiv \frac{2}{\sqrt{\pi}}\int_x^{\infty} \exp\left(-u^2\right)du$$

Rewriting (6.3) to solve for Th gives us

(6.6) $$Th = \sqrt{2N_0{}^2TW}\ \mathrm{erfc}^{-1}(2P_{fa}) + N_0TW$$

and putting this in (6.4) yields

(6.7) $$P_d = \tfrac{1}{2}\mathrm{erfc}\left[\frac{\sqrt{2N_0{}^2TW}\,\mathrm{erfc}^{-1}(2P_{fa}) - \varepsilon_c}{\sqrt{2N_0{}^2TW + 4N_0\varepsilon_c}}\right]$$

Now, we will assume the noise energy collected by the radiometer, $N_0TW$, greatly exceeds the signal energy collected, giving us $2N_0{}^2TW >> 4N_0\varepsilon_c$ which allows a simplification of the denominator in the argument of the error function in (6.7), or

(6.8) $$P_d \approx \tfrac{1}{2}\mathrm{erfc}\left[\frac{\sqrt{2N_0{}^2TW}\,\mathrm{erfc}^{-1}(2P_{fa}) - \varepsilon_c}{\sqrt{2N_0{}^2TW}}\right] = \tfrac{1}{2}\mathrm{erfc}\left[\phi(W)\right]$$

where

(6.9) $$\phi(W) = \mathrm{erfc}^{-1}(2P_{fa}) - \frac{\varepsilon_c}{\sqrt{2N_0{}^2TW}}$$

We want to maximize $P_d$ with respect to W. To do this, we take the partial derivative of (6.8)

(6.10) $$\frac{\partial P_d}{\partial W} = \frac{\partial P_d}{\partial \phi(W)}\frac{\partial \phi(W)}{\partial W}$$

and set it equal to zero. Looking at the first product on the right hand side we have

$$\frac{\partial P_d}{\partial \phi(W)} = \frac{1}{2}\frac{\partial}{\partial \phi(W)}\operatorname{erfc}\left[\phi(W)\right] = \frac{1}{2}\frac{\partial}{\partial \phi(W)}\left[\frac{2}{\sqrt{\pi}}\int_{\phi(W)}^{\infty}\exp(-u^2)du\right]$$

(6.11)

$$= -\frac{1}{\sqrt{\pi}}\exp\left[-\phi(W)^2\right] \neq 0$$

so we need to find out where the second product on the right hand side of (6.10) goes to zero

(6.12) $$\frac{\partial \phi(W)}{\partial W} = \frac{\partial}{\partial W}\left[\operatorname{erfc}^{-1}(2P_{fa}) - \frac{\varepsilon_c W^{-1/2}}{\sqrt{2N_0{}^2}}\right] = -\frac{1}{\sqrt{2N_0{}^2}}\left[W^{-1/2}\frac{\partial \varepsilon_c}{\partial W} - \frac{1}{2}\varepsilon_c W^{-1/2}\right] = 0$$

or

(6.13) $$\frac{\partial \varepsilon_c}{\partial W} - \frac{\varepsilon_c}{2W} = 0$$

From (6.2) we find

$$\frac{\partial \varepsilon_c}{\partial W} = \frac{\partial W/2B}{\partial W}\frac{\partial \varepsilon_c}{\partial W/2B} = \frac{1}{2B}\frac{\partial}{\partial W/2B}\left[2E\int_0^{W/2B}\operatorname{sinc}^2(y)dy\right]$$

(6.14)

$$= \frac{E}{B}\operatorname{sinc}^2\left(\frac{W}{2B}\right)$$

So, using (6.2) and (6.14), (6.13) will be satisfied when

(6.15) $$\operatorname{sinc}^2\left(\frac{W}{2B}\right) = \frac{B}{W}\int_0^{W/2B}\operatorname{sinc}^2(y)dy$$

So we see that, because of our assumption about the signal energy level, the best value for W will essentially depend only on B. The solution to both sides of (6.15) with respect to W/B are shown in Figure 6.6, and we see

(6.16) $$1.00 < \frac{W}{B} < 1.05$$

For our purposes we will accept $W \approx B$ as the answer we seek.

As an indication of the sensitivity of W in determining $P_d$, Figure 6.7 plots these two variables using (6.7) (not the approximation), for a specific example in which $P_{fa} = 0.1$, $T = 32768$ seconds (normalized) and the signal to be detected is DS spread with $B = 0.015625$ Hz and $E = 129$. As we see, good results can be obtained for a wide range of W, even up to about 2B.

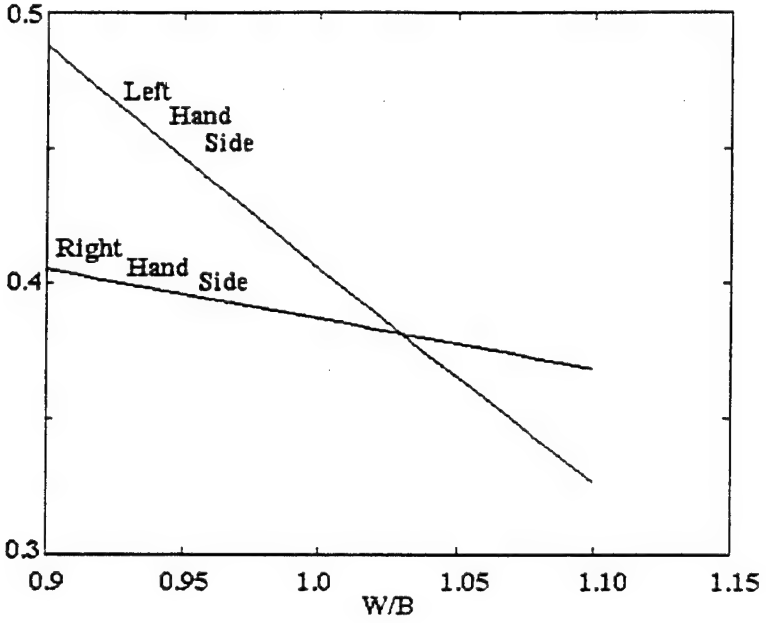



Figure 6.6. Solution to Both Sides of Equation (6.15)

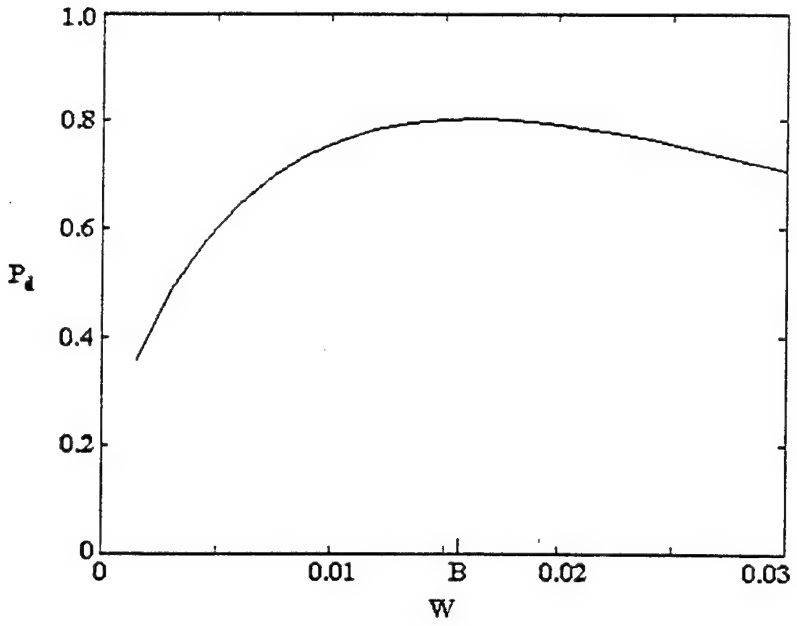



Figure 6.7. Effect of Filter Size on the Probability of Detection

**Detection When the Signal Parameters Are Known**

We now consider detection when the interceptor knows the signal's center frequency and bandwidth. The interceptor, therefore, knows exactly where to look in the time frequency plane for signal energy. In this sub-section we will examine how to use the QMF bank tree as a radiometer detector of these signals, and look at a simulation. The main purpose for this is to provide a check on our analysis to this point. However, since the next step in the chapter will be to consider detection of DS signals when the center frequencies and bandwidths are not known, the results here will also provide a convenient best case to compare those results against.

From our analysis above, we saw that a radiometer detector of the form shown in Figure 6.5 will be most effective when $W \approx B$. Let's now consider a case where we set W exactly equal to B, and let

$$\begin{aligned} & W = B = 0.015625 \text{ Hz} \\ & T = 2^{15} = 32768 \text{ sec} \\ & N_0 = 2 \\ & \varepsilon_c = 100 \end{aligned} \tag{6.17}$$

Using (6.7) and the values given in (6.17), we can plot a theoretical receiver operating characteristic (ROC) curve, and this is shown as the solid line in Figure 6.8.

In the simulations, a waveform was decomposed using the QMF bank tree to layer five. (At this layer, the tile width, in the frequency dimension, is equal to the values given in (6.17) for W and B.) The energy in the layer five tiles were added as shown in Figure 6.1 to yield a 32 element spectral vector. To obtain values for $P_{fa}$ WGN alone was used for the waveform. To obtain values for $P_d$ the same noise was used with a DS signal added. The center frequency of this signal was set equal to the center frequency for a particular spectral vector bin, and, to make the detection decision, the energy in this bin was compared against various threshold values.

One hundred runs were made for each of three sets, with a signal energy of 100 and center frequency of 0.3359375 Hz used in every case. Results of these simulations are shown (with X's) plotted on Figure 6.8. As we can see, the agreement between the theoretical results and the simulations are very good.

Another three sets of simulation runs were carried out with a DS signal center frequency of 0.2578125 Hz (that is, closer to the center frequency of the spectral vector). The results of these are also shown (with o's) in Figure 6.8, and we can see they are not as good. The reason for this is due to the non-ideal nature of the filters used in the QMF bank tree--even though 32 coefficient modified sinc filters were used. The DS signal was located near the center of the time frequency plane, and therefore, some of its energy was sent into the low pass filter in the initial layer of the tree and showed up in an adjacent bin.

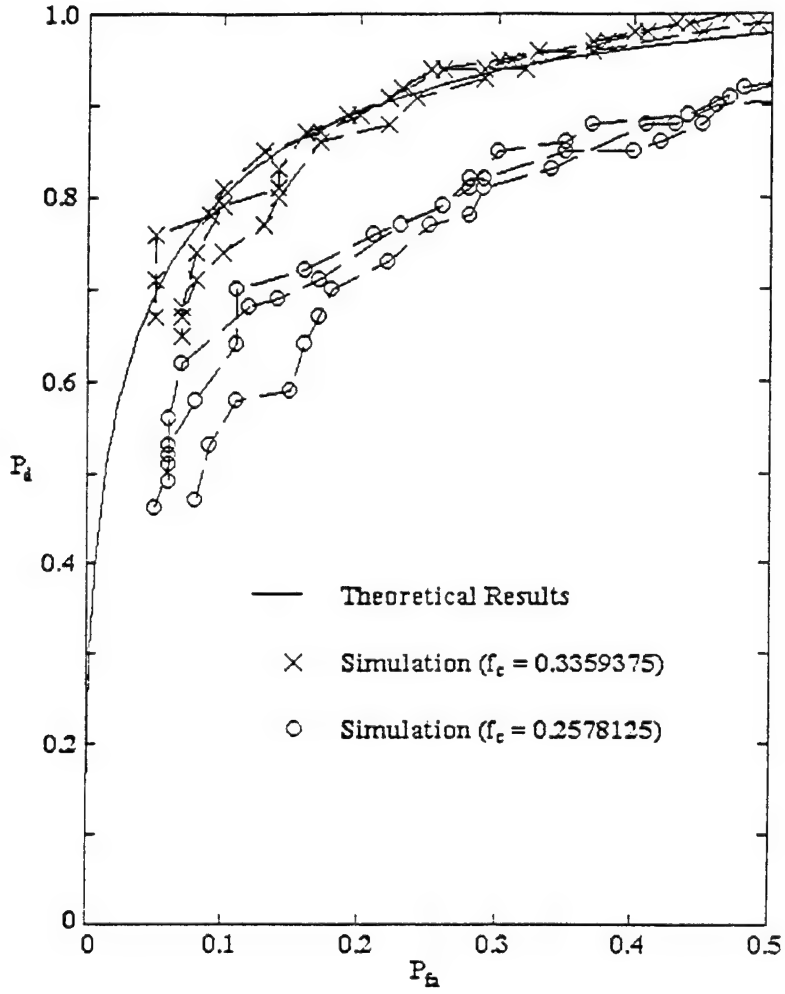



Figure 6.8. Results When the DS Signal Parameters Are Known

**Detection With Unknown Signal Parameters**

Let's now relax the requirement that the interceptor knows the center frequency of the DS signal to be detected. (For the moment, we will still assume the interceptor knows the signal's bandwidth). Obviously, if the interceptor simply looked at the spectral vector whose bin width was equal to the signal bandwidth, the signal and QMF bank tree tiles would not, in general, be aligned in frequency. Hence, less signal energy would be collected, and detection would suffer.

A better idea is to decompose the signal further through additional layers of the QMF bank tree, and obtain a higher resolution spectral vector. Then, the energy from certain adjacent bins can

be summed, and the result can be compared to a threshold. Since the interceptor is assumed to know the signal's bandwidth, the number of adjacent bins to be summed should be whatever number times the bin bandwidth most closely equals the signal bandwidth. Because this is a sort of two dimensional version of a "block" introduced in Chapter V, we will call it a "rectangle" here. Since the center frequency of the signal is unknown, all positions of the rectangle in the spectral vector should be examined.

This search is particularly easy to carry out, once we note that it is a convolution between the spectral vector and a rectangular window. If our goal is simply to determine whether one or more signals are present, we can take the largest value from the convolution results, and compare that to a threshold. (We expect this will be the value from the point where the rectangle is most nearly centered over the signal.)

A modification of this idea, reminiscent of a similar algorithm described in Chapter V, is to first look for the highest energy value from the convolution result, save the result and its position to a "rectangle list", then throw out adjacent values from overlapping rectangles, repeating the procedure to find a list of potential DS signal locations. This list could, in the case of a receiver like the one described in Figure 1.2, then be fed to a classifier whose job is to determine which, if any, of the positions contain DS signals. With this algorithm, multiple DS signals can be detected, so long as they do not overlap in frequency.

Now we further relax our requirements and assume the interceptor does not necessarily know the DS signals' bandwidths. An obvious modification to the algorithm is to simply repeat the procedure for different sized rectangles. As we saw above, the results from the rectangles closest in size to a signal's bandwidth should yield the best probability of detection. Of course, other sized rectangles will also pick up the signal's energy. What we want to do is to find those results yielding the best probability of detection, saving them to the rectangle list, and then discard overlapping results from different sized rectangles.

This leads to a question: Given the results of two different sized rectangles, both indicating a DS signal in a particular location, how do we decide which to save? We can't use energy, because the larger rectangle will collect both more signal energy and noise energy. Instead, we go back to (6.8) and note we can increase $P_d$ by decreasing $\phi(W)$. This is the same as increasing the second term on the right hand side of (6.9), which, in turn is the same as increasing the test statistic

$$u \equiv \frac{\varepsilon_c(W)}{\sqrt{W}} \tag{6.18}$$

Our expected observed energy, $\varepsilon_o$, is

$$E[\varepsilon_o] = E[\varepsilon_c] + N_0TW \tag{6.19}$$

so we can find a value for u from

$$u = \frac{\varepsilon_o - N_0TW}{\sqrt{W}} \tag{6.20}$$

What we should do, then, is find a list of high energy rectangles, as described above, for each rectangle size, and compute (6.20) for each. We then save the highest value of u, throwing out results from rectangles that overlap, and repeat, until the final rectangle list contains u, the energy, and the

positions of non-overlapping rectangles which may contain DS signals. To make a detection decision, we will compare the largest value of u found against a threshold.

Note that (6.20) could also be used to make the selection among overlapping rectangles of the same size. Since W will be identical for the competing rectangles, there is no difference between comparing values of u and comparing values of $\varepsilon_o$.

Just as with overlapping blocks in Chapter V, This algorithm cannot be fully analyzed theoretically because the overlapping rectangles are selected based partly on shared energy. Again, we are confronted with dependent order statistics. We can, however, use the case of a known DS signal position and bandwidth as a best case, and compare our simulation ROC results against these.

Matlab code to carry out and simulate this algorithm is listed and described in Appendix C.

### Analysis Results

The detection simulations are similar to the ones described above where the signal's center frequency and bandwidth are known to the interceptor, except here the DS signal's center frequency in each run is randomly set to a value between 0.125 + B/2 and 0.375 - B/2 Hz, and the spectral vector is formed from the eighth layer output of the QMF bank tree. In these simulations, the signal's bandwidth and energy were identical to those used in the simulations above.

Three sets of 100 runs each were made, and the results are shown in Figure 6.9, along with the theoretical curve from Figure 6.8. As expected, the results are not as good as when the interceptor knows the center frequency and bandwidth of the signal to be detected.

## Feature Extraction

Clearly, the algorithm we developed above for detection can also be used to estimate features of the DS signal. We are interested in the signal's bandwidth, center frequency, and energy. In this section we will first discuss the details of making these estimates, and then show simulation results verifying the effectiveness of the methods.

### Bandwidth

We saw above that our best probability of detecting a signal occurs when the size of the rectangle approximately equals the signal's bandwidth. By turning this logic around, we see the size of the rectangles output from the algorithm can be used as an estimator of the signal's bandwidth.

We should not expect too much from this estimator, however. Just as Figure 6.7 tells us we will still be able to detect a signal when the rectangle doesn't perfectly match the signal's bandwidth, so too does it suggest we will suffer a wide variance in our estimate. Also, we have not yet examined how an offset of the signal's center frequency, relative to the tiling of the time frequency plane, affects detection, or this estimate.

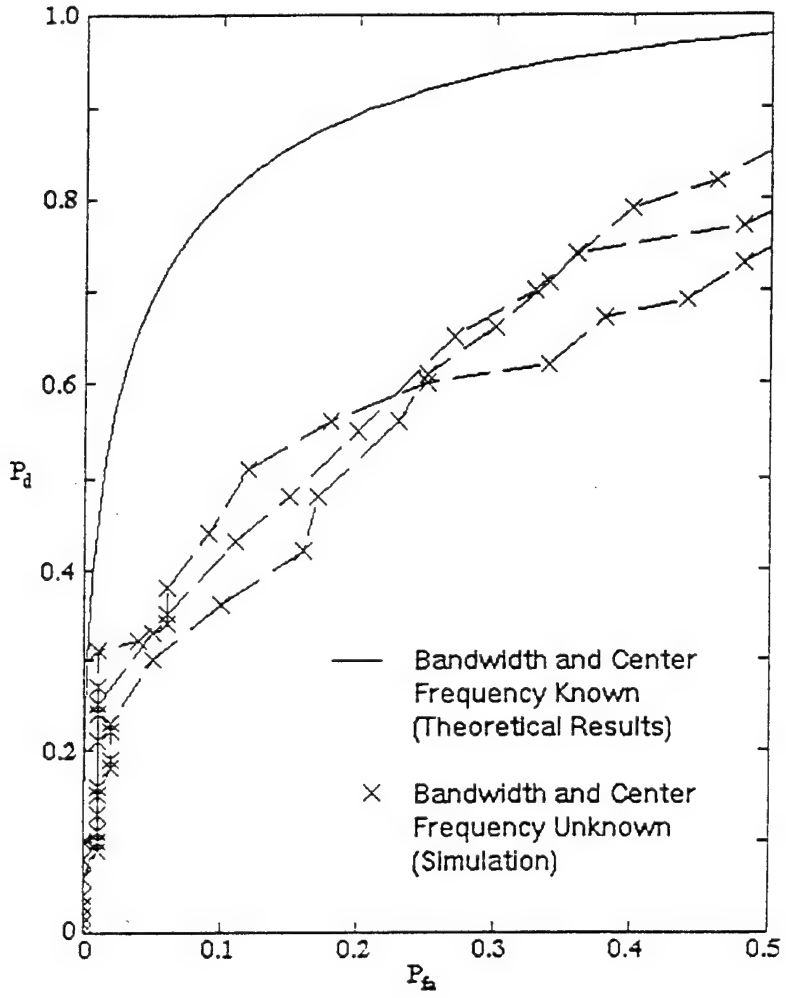



Figure 6.9. Analysis Results When the Signal's Center Frequency and Bandwidth Are Unknown

To examine this, let's consider a specific example--the one we've been using throughout the chapter--where the signal's bandwidth is B = 0.015625 Hz, and where we decompose the waveform to the eighth layer of the QMF bank tree to give us a spectral vector with a bin size of 1/512 Hz. Figure 6.10 shows this, with rectangles equal to seven, eight, and nine bins, superimposed on energy distributions of the DS signals with the best (solid curve), and worst (dashed curve) possible alignments. The amount of signal energy collected using our algorithm is, in each case, proportional to the area of the sinc squared curve covered by the rectangle. As we can see, for the eight bin rectangle the worst case alignment occurs when the signal's center frequency is located at the center of a bin, while, for the odd sized rectangles, the worst case alignment occurs when the signal's center

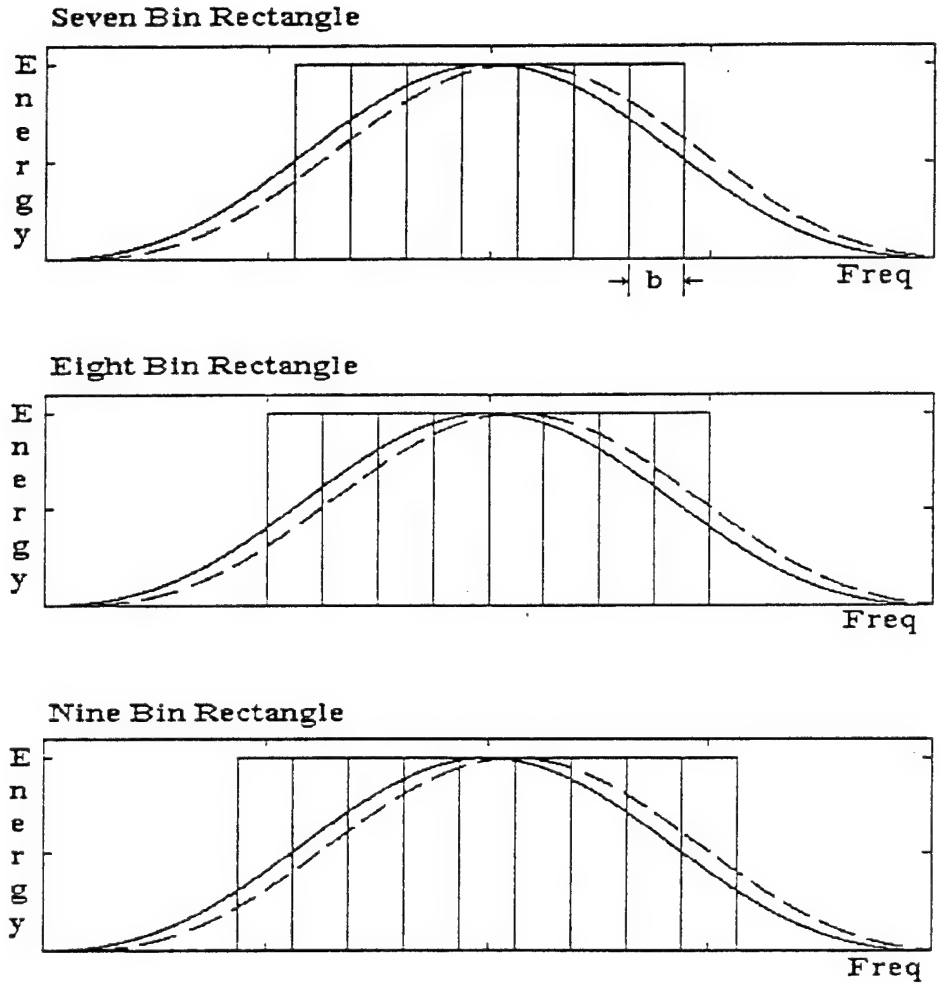



Figure 6.10. Seven, Eight, and Nine Bin Rectangles Superimposed on Sinc-Squared Curves to Give an Indication of How Much DS Signal Energy Each Will Collect. The Solid Curves Show Signals Centered Under the Rectangle. The Dashed Curves Show Signals as Far Off Center as Possible.

frequency is located at the edge of a bin. (If the signal's center frequency were shifted any further than shown in the figure, the bin on the left of the existing rectangle could be dropped off, a new bin on the right added, and this new rectangle would collect more energy.)

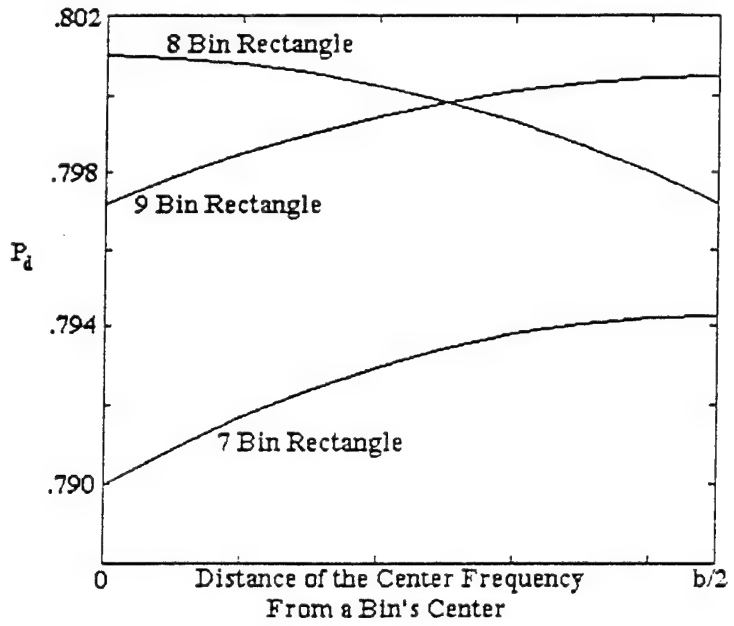



Figure 6.11. Probability of Detection For Seven, Eight, and Nine Bin Rectangles, as the Signal's Center Frequency is Shifted With Respect to a Bin's Center, (b ≡ Bin Bandwidth)

It is easy to compute the amount of signal energy, $\varepsilon_c$, collected with respect to various alignments. When those values for $\varepsilon_c$, and a $P_{fa}$ of 0.1, are used in (6.7), we find the curves shown in Figure 6.11. As we can see, when the center frequency is far enough from the edge of a bin, a length nine rectangle will actually have a slightly greater probability of detection. However, the difference in $P_d$ is actually very slight, and, in fact, is undetectable in our simulations. As we will see below, the effects of the noise added to the signal will be the greater influence. (Although we don't explore the possibility here, it seems likely that weighting the bins to obtain a window with a shape different from rectangular--perhaps one more closely matched to the energy distribution of the DS signal--may improve the bandwidth estimate, possibly at the expense of $P_d$ in the detection problem. This is discussed further in Chapter VIII.)

### Center Frequency

The center frequency of the DS signal is very easy to estimate. It is simply the center frequency of the rectangle found by our algorithm. When the signal is correctly detected, we would expect a maximum error in this value to be equal to half of the bin size, or, in our simulations using layer eight, 1/1024 Hz.

### Signal Energy

Assuming we have correctly detected a signal, the pdf for the energy collected, $\varepsilon_o$, should be a Chi-Squared curve with a non-centrality parameter equal to the amount of signal energy collected. Since we are dealing with large time bandwidth products, we can, as we did earlier in the chapter, approximate the Chi-Squared pdf with a Gaussian pdf.

Rewriting (6.19) we obtain

$$E[\varepsilon_c] = E[\varepsilon_o] - N_0TW \tag{6.21}$$

where W is equal to the size of the rectangle (in Hertz) found using the algorithm. We conclude, therefore, that we can obtain an estimate of the signal energy with

$$\varepsilon_c = \varepsilon_o - N_0TW \tag{6.22}$$

and, because we are dealing with approximately Gaussian statistics, the distribution of $\varepsilon_c$ will be approximately Gaussian.

### Analysis Results

The Matlab code to simulate feature extraction is listed and described in Appendix C. Five sets of runs were made for this section. In each succeeding set the signal energy was increased as shown in Table X.

**Table X**

Signal Energy and Amplitude For the DS Signal Feature Extraction Simulations

| Set Number | $\varepsilon_c$ | $E = \varepsilon_c/0.774$ | $A = \sqrt{2E/T}$ |
|---|---|---|---|
| 1 | 100 | 129 | 0.0887 |
| 2 | 200 | 258 | 0.1255 |
| 3 | 400 | 517 | 0.1773 |
| 4 | 800 | 1034 | 0.2512 |
| 5 | 1600 | 2067 | 0.3553 |

The results of the bandwidth estimates are shown in Figure 6.12 as histograms, with each bar representing a particular rectangle size in numbers of bins. The actual DS signal bandwidth in the simulations was equal to eight bin widths. As we see, the estimate improves for higher signal energies, but even for set five we have a fairly wide variance.

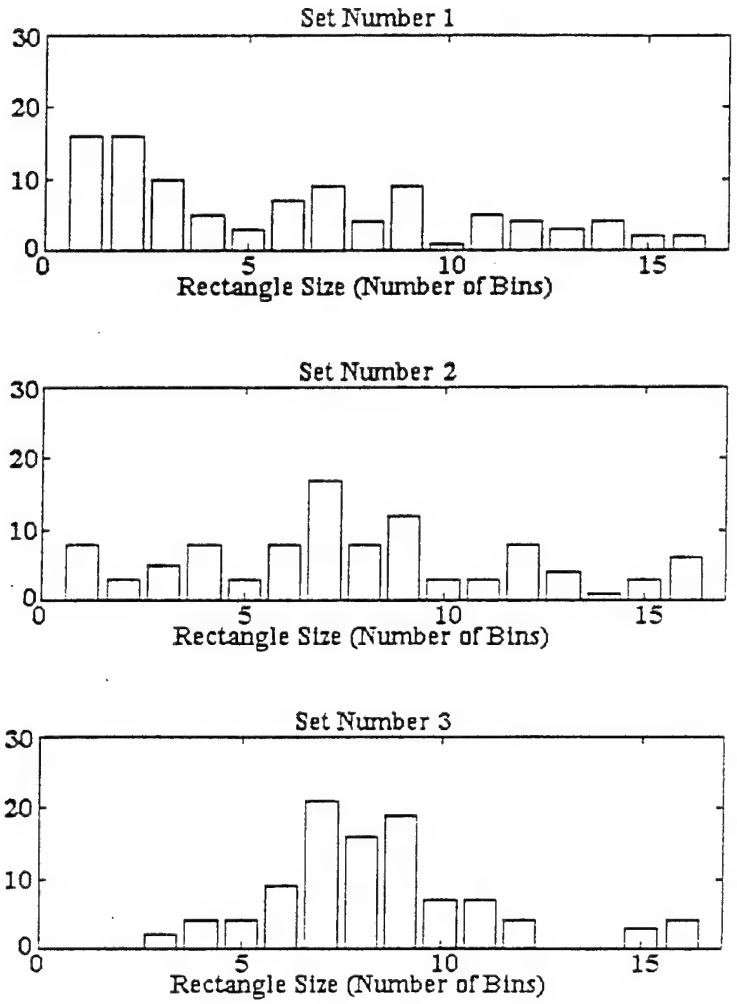



Figure 6.12a. Rectangle Sizes Found
Vertical Axis Represents the Number Found

The mean and variance of error in the center frequency estimate for each set are listed in Table XI. The numbers suggest that after the signal energy is raised beyond about $\varepsilon_c = 400$ no further improvement can be expected. The remaining error is due primarily to the resolution of the tiling. The mean and standard deviation of the signal energy estimates, made using (6.21), are also shown in Table XI.

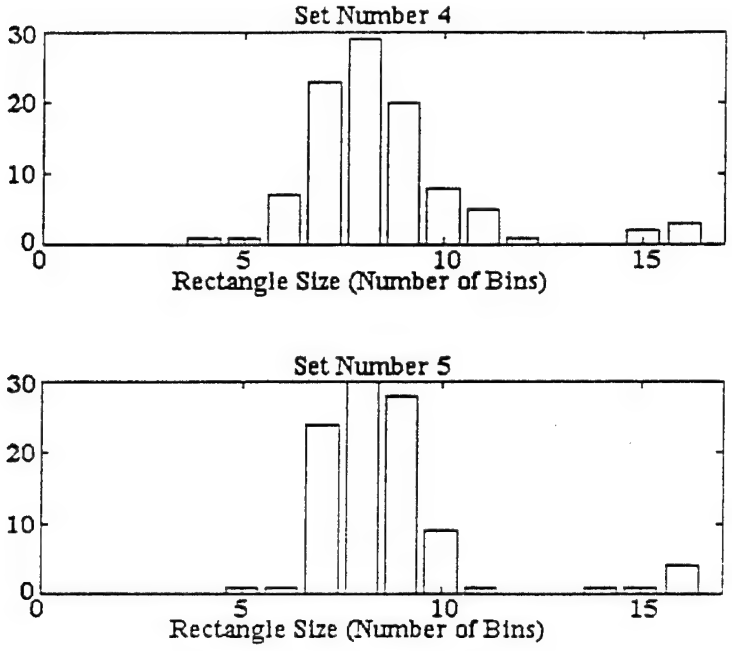



Figure 6.12b. Rectangle Sizes Found
Vertical Axis Represents the Number Found

Table XI

Error in the Center Frequency and Energy Estimates in the DS Signal Feature Extraction Simulations

| Set Number | Center Freq Error | | Signal Energy Estimate | |
|---|---|---|---|---|
| | mean | variance | mean | std dev |
| 1 | $1.15\times10^{-2}$ | $4.50\times10^{-3}$ | 136 | 66.016 |
| 2 | $3.40\times10^{-3}$ | $1.20\times10^{-3}$ | 223 | 87.036 |
| 3 | $1.77\times10^{-5}$ | $5.93\times10^{-6}$ | 417 | 89.566 |
| 4 | $1.84\times10^{-4}$ | $4.46\times10^{-6}$ | 802 | 105.820 |
| 5 | $1.64\times10^{-4}$ | $4.56\times10^{-6}$ | 1591 | 146.423 |

## Summary

In this chapter we began by describing the characteristics of the DS signal that we are looking for. Right away, we concluded that the decomposition by the QMF bank tree is more mathematically intensive, but provides no more information, than a similar FFT, and therefore should not be performed exclusively to obtain the information we are looking for here. If the decomposition

is needed for other reasons, however, such as the detection of other types of spread spectrum signals, or for an exploration of the changes in internal energy distribution in the DS signal, it can also be used for DS signal detection, and for the estimation of the DS signal's energy, center frequency, and bandwidth.

We then looked at the detection of these signals when the center frequency and bandwidth were known, and found ROC curves by simulation that matched the theoretical radiometer detection curve, except for a small loss due to the non-ideal nature of the QMF filters. Then, an algorithm was developed to detect DS signals when the center frequency and bandwidth are not known to the interceptor, and simulations were performed to obtain ROC curves.

Finally, we saw how the information obtained from the detection algorithm can be used to estimate the DS signal's energy, center frequency, and bandwidth. In a QMF bank tree intercept receiver like that shown in Figure 1.2, this information could be sent as a list from the analyzer block to the classifier block. In this architecture, all of the possible signals found by the algorithm would be listed and it would be the classifier's job to determine which, if any, were DS signals.

There is no reason the algorithm described in this chapter cannot be used at the same time as the algorithm described in Chapter V for the detection and feature extraction of hopped signals with time bandwidth products of one. Signals of that sort are unlikely to cause a false detection in the DS signal detection algorithm because the hops are spread across the time frequency plane and little signal energy is likely to be concentrated at any particular band of the spectral vector. Likewise, DS signal energy is too spread in the time frequency plane to be likely to cause a detection in the Chapter V algorithm, which looks for relatively high concentrations of energy within small areas of the plane.

## VII. Interception of Fast FH/DS, TH/DS, Fast FH/TH/DS, and Slow FH/DS Signals

### Introduction

In this chapter we look at the detection and feature extraction of hopped signals whose cells have time bandwidth products greater than unity. These include signals whose cells have been spread using DS techniques (these will be referred to collectively, in this chapter, as "hopped/DS" signals), and also Slow FH signals where the spreading of cell energy is due to the FSK modulation of the carrier by information.

As in past chapters, we will use "detection" to refer to the process, by the interceptor, of deciding whether one or more signals are present in a waveform. "Feature extraction" will refer to estimating the cells' energy, position in the time frequency plane, and dimensions in both time and frequency.

We begin the chapter with a short description of the characteristics of the signals. Then we discuss the general problem of detecting these signals. Since much of this problem parallels the problems of detecting hopped signals with unity time bandwidth products and detecting DS signals, we will often refer to results deduced in Chapters V and VI.

We will then develop an algorithm specifically to detect the signals of interest in this chapter. Its implementation will be discussed and the results presented.

Finally, we will see how the results of the detection algorithm, together with a modification of the techniques discussed in Chapter VI, can be used to estimate the signal features we are interested in.

### Hopped Spread Spectrum Signals With Time Bandwidth Products Greater Than One

#### Hopped/DS Signals

These signals are similar in structure to the Fast FH, TH, and Fast FH/TH signals discussed in Chapter V, except that each cell, rather than consisting of a single tone, will be formed from a tone modulated by a random binary waveform. The effect is that each cell will look like a short time duration version of the DS signal discussed in Chapter VI.

The transition duration of the random binary waveform (1/B in Figure 6.3) should be much smaller than the cell duration, so the cell's energy will be distributed across the frequency band (approximately) with a sinc squared distribution as shown in Figure 6.4, where $f_c$ is the cell's tone frequency. The random binary waveform spreads the energy but does not affect the total cell energy. As with unity time bandwidth cells, the cell energy is

$$\varepsilon = \frac{A^2 T}{2} \tag{7.1}$$

where A is the signal's amplitude and T is the cell duration.

### Slow Frequency Hopped Signals

The Slow FH signal structure has been described briefly in Chapter IV. Basically, the information rate is faster than the hop rate, and the information is modulated using FSK, resulting in a cell with energy concentrated as shown in Figure 4.7. In the discussion that follows, we will refer to the gray shaded regions in Figure 4.7 as "micro-cells" to distinguish them from the larger hop cell.

Each micro-cell has a time bandwidth product of one. Since we assume the transmitter turns each micro-cell on, and later off, instantaneously, the energy distribution in the frequency dimension will be sinc squared with a bandwidth, bw, equal to the inverse of the micro-cell's duration. When the signal structure is designed so there are many micro-cells in each cell, there will be approximately equal numbers in each information channel, and the cell's energy distribution across the frequency dimension will look something like the solid line in Figure 7.1. If, on the other hand, there are only a few micro-cells in each cell, it is likely that only a few of the available information channels will be filled in each cell, resulting in different energy distributions that will change drastically from cell to cell in a single signal. This latter case is the one we will confront in our simulations. Regardless of the distribution, the cell energy for the slow FH signal is given by (7.1).

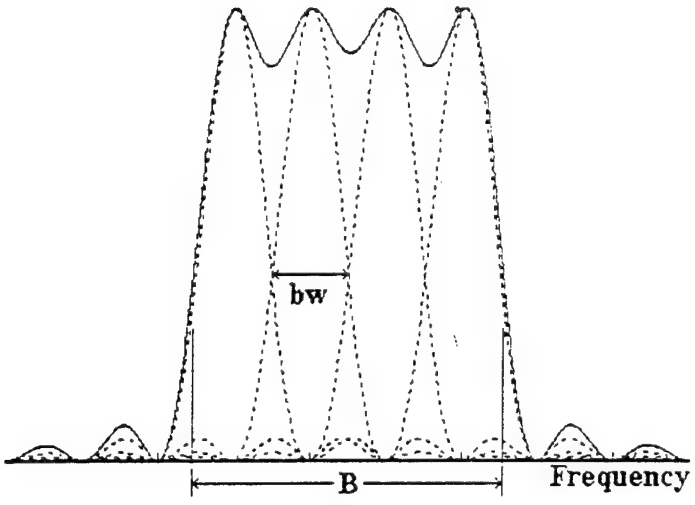



Figure 7.1. Energy Distribution in a Four Channel Slow Frequency Hopped Cell

### Detection

The detection of the signals of interest in this chapter, when only one signal is present at a time, and when the interceptor knows the hop channelization, hop duration, and hop timing of the signals to be detected, has been widely discussed in the literature; and this discussion generally

parallels that presented for signals with unity time bandwidth cells [15] [16] [18] [19] [30] [35] [41] [42] [45]. When the interceptor knows those signal features, and also the intercepted cell energy, the optimal detector is the one presented in Figure 5.5, except the bandpass filters in the bank of radiometers will be tuned to the wide bandwidth hop channels, and equation (5.10) will be replaced with [16]

$$L = \sum_{j=1}^{M} \frac{I_{TW-1}\left(\frac{1}{\sigma^2}\sqrt{x_j \varepsilon}\right)}{\left(\frac{1}{\sigma^2} x_j\right)^{(TW-1)/2}} \tag{7.2}$$

where

| | |
|---|---|
| TW | is the cells' time bandwidth product |
| $I_{TW-1}(\bullet)$ | is the modified Bessel function of the first kind, TW-1 order |
| $\varepsilon$ | is the cell energy |
| $\sigma^2$ | is the noise variance |
| M | is the number of hop channels |

(Note that when TW equals one (7.2) simplifies to (5.10).)

When only one signal is present, and the interceptor knows all of the signal features listed above, except for the intercepted cell energy, the FBC shown in Figure 5.6 is a good detection architecture to use. As above, the input filters of the radiometer bank should be tuned to match the wide bandwidth hop channels.

Our main interest here, of course, is the case where the interceptor does not know, a priori, the amount of cell energy intercepted, hop channelization, hop duration, or hop timing. As we saw in Chapter V, if we knew nothing about the signal structure a single radiometer covering the region, with a threshold detector, would be the best architecture. Just as with the signals in Chapter V, however, the signals of interest in this chapter have one important characteristic: Their energy is concentrated in cells. We might, therefore suppose some sort of detection algorithm using overlapping blocks in the time frequency plane, similar to the "nine tile scheme" laid out in Chapter V, could offer some sort of improvement.

The next question, then, is: How large should our blocks be? In Chapter V, we used three by three blocks because (using multiple layers of the QMF bank tree as necessary) we could be assured a particular block covered most of each cell's energy. This, of course, depended on our knowledge of the cell time bandwidth product. For the signals of interest in this chapter, however, we do not have that information.

Ideally, we would like our block dimensions to match the cell dimensions. (We will see this demonstrated in an example below.) Also, because the cell time bandwidth products are so large compared to our QMF bank tree tiles, the relative alignment is less significant than for the signals in Chapter V, and we can assume that any energy falling into tiles outside of the block is insignificant.

One possibility is to use an intermediate layer (where each of the tiles' time and frequency dimensions is certain to be less than the corresponding cells' dimensions) of the QMF bank tree, and exhaustively search the space using blocks of every possible dimension. This does, however, increase

the computational burden significantly. A better idea is to determine how far our blocks can be from the ideal and still have a high probability of detecting a cell. We could then limit our algorithm to a single pass with a compromise block size, or if necessary (although we will not explore this further) use several passes with widely different block dimensions.

This discussion appears to be analogous to that in Chapter VI, where we determined how the rectangle size (in frequency) affects the probability of detection of a DS signal. We used the Gaussian approximation for the Chi-Squared pdf's with large numbers of degrees of freedom [41] and derived Equation (6.7), which we rewrite here for convenience (altering the notation slightly)

$$P_d = \tfrac{1}{2}\operatorname{erfc}\left[\frac{\sqrt{2N_0^2 t_0 W}\,\operatorname{erfc}^{-1}(2P_{fa}) - \varepsilon_c(t_0,W)}{\sqrt{2N_0^2 t_0 W + 4N_0\varepsilon_c(t_0,W)}}\right] \tag{7.3}$$

This will give us the probability of detection of a single cell of a signal with large time bandwidth cells when

| | |
|---|---|
| $t_0$ | is the block's time duration |
| $W$ | is the block's bandwidth |
| $\varepsilon_c(t_0,W)$ | is the amount of cell energy collected in the block |
| $N_0$ | is, as before, the noise energy density (usually normalized to two) |
| $P_{fa}$ | is the probability of false alarm due to noise energy collected in the block |

The amount of cell energy collected will, of course, depend on the block size. Let us first assume we have a hopped/DS cell, and the block is centered over the cell. (This latter assumption will be approximately valid in our algorithm because the difference between tile and cell size will mean there will always be a block close to this alignment.) If we assume the cell energy is uniformly distributed in time across the cell's duration, and consider a block that covers the entire frequency dimension but may, or may not, cover the cell's duration in time, the energy collected will be

$$\varepsilon_c(t_0) = \varepsilon \min\{1, t_0/T\} \tag{7.4}$$

where T is the cell's duration. If, on the other hand, we have a block that covers the entire observation interval, the amount of cell energy depending on the block's bandwidth will yield

$$\begin{aligned}\varepsilon_c(W) &= \frac{\varepsilon}{B}\int_{-W/2}^{W/2}\operatorname{sinc}^2(x/B)dx = \frac{2\varepsilon}{B}\int_0^{W/2}\operatorname{sinc}^2(x/B)dx \\ &= 2\varepsilon\int_0^{W/2B}\operatorname{sinc}^2(y)\,dy\end{aligned} \tag{7.5}$$

where B is the bandwidth of the cell. Combining (7.4) and (7.5) gives us

$$\varepsilon_c(t_0,W) = 2\varepsilon \min\{1, t_0/T\}\int_0^{W/2B}\operatorname{sinc}^2(y)\,dy \tag{7.6}$$

which is, of course, the amount of cell energy collected by a block with dimensions $t_0$ by W, centered over the cell.

Using (7.3) and (7.6) we can construct an example to explore the effect of changing the block size on our ability to detect a cell. In this example, we use a constant $P_{fa}$ of 0.1, and a cell size of B = 0.0611 Hz, T = 1637 sec, with $\varepsilon$ = 129 and $N_0$ = 2. Figures 7.2 and 7.3 both show the contour plot of $P_d$ as the block size is adjusted. Figure 7.2 presents the plot on a linear scale comparing $t_0/T$ and W/B. Figure 7.3 shows the same, but uses a base ten log-log scale. This plot is particularly useful for answering questions like: Which is better, a block that is twice as large as the cell in each dimension, or one that is half as large? Apparently, the answer is that they are about the same.

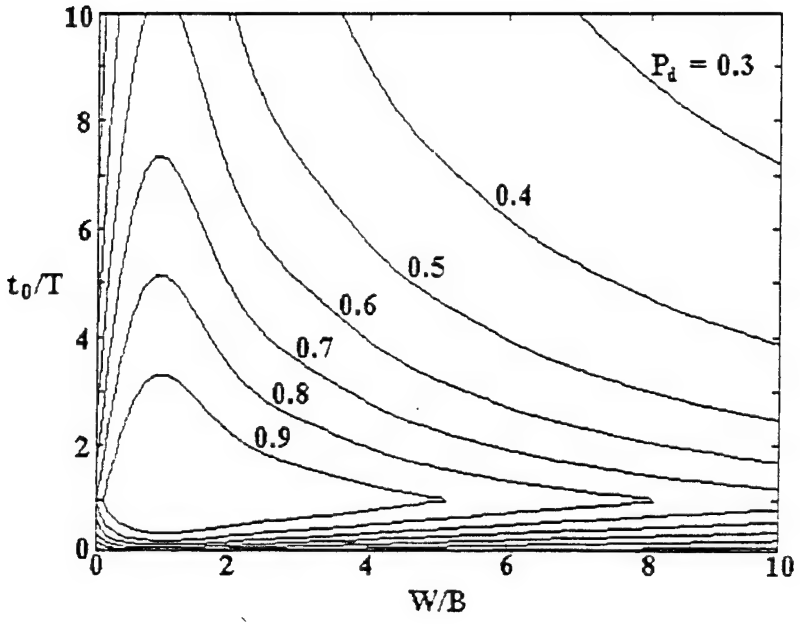



Figure 7.2. Probability of Detection of Hopped/DS Cell

Recall, however, that we found (7.4) by assuming the energy distribution was uniform in time. If we use a block that is larger than the cell in time, the equation would give us $\varepsilon_c = \varepsilon$ regardless of the validity of that assumption. Similarly, we assumed we had a hopped/DS cell and based (7.5) on the energy distribution in frequency for that type of cell. If we use a block that is larger than the cell in frequency we will collect most of the cell energy regardless of the distribution and will achieve similar results for a Slow FH cell.

Therefore, for our detection algorithm, we should determine the range of possible cell sizes that we wish to detect and begin by using blocks that match the largest dimensions in both time and frequency. If, based on an analysis similar to the one producing Figure 7.3, that block size will yield an unacceptably low $P_d$ for smaller cells, more passes with smaller blocks may be made.

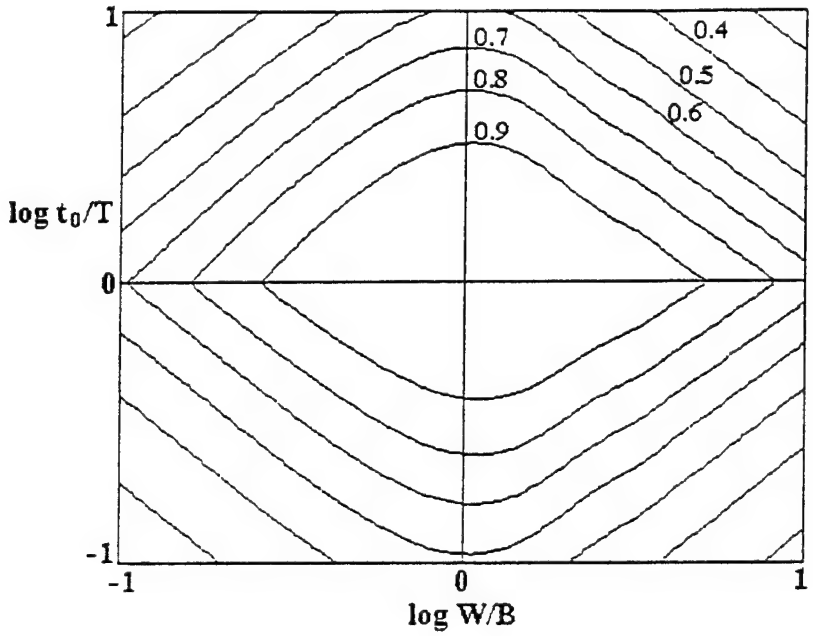



Figure 7.3. Probability of Detection of Hopped/DS Cell (Base Ten Log-Log Scale)

Notice the best block size is the one that matches the cell size, confirming a statement made earlier.

To complete the algorithm: After obtaining the amount of energy in each overlapping block, the blocks can be sorted, from greatest to least, and, just as with the nine tile scheme, the block with the largest energy can be saved, overlapping blocks discarded, with the process repeated until a list of non-overlapping blocks is obtained which will, for the most part, contain the signal(s) cells. As we will see later, we can then look at the energy distribution in each block to estimate the cell's position, dimensions, and energy.

**Analysis**

As we saw in Chapter V, there appears to be no mathematically tractable way to analyze a scheme that looks for concentrations of energy among overlapping blocks. Our goal, then is to use simulation to verify that there are cases in which the block algorithm can produce better results than the default--a radiometer detector covering the entire observation interval, and with the input filter tuned to cover a signal's total bandwidth. (This case is, of course, mathematically tractable: The ROC curve can be generated from (7.3) when the collected cell energy is replaced by the total collected signal energy, $t_0$ replaced by the radiometer's observation interval, and W replaced by the radiometer's bandwidth.)

To simplify the simulations somewhat, we will consider detection when only a single signal cell is present in the observation region. This simplifies the implementation of the block algorithm

significantly, as we only need to concern ourselves with the largest energy block, and not with discarding overlapping blocks. It is also easy, in this case, to find an upper bound on our detection capabilities. To do this, we assume the interceptor knows where to look for the cell, and compute an ROC curve from (7.3).

**Hopped/DS Results.** Six sets of runs were made to simulate the detection of a hopped/DS (specifically a TH/DS) signal. The 32 coefficient modified sinc filter was used for the first three, and the 22 coefficient energy concentration filter was used for the other three. The signal cells were 818 seconds long (normalized) and had a bandwidth of 0.0305 Hz. The signal amplitude was set to 0.3496 which equates to a cell energy of 50.

As a validity check, in addition to determining data for the block algorithm, the simulation code also finds experimental ROC values for a radiometer covering the entire region of interest in the time frequency plane, and for a radiometer matched to the cell's position. These results are shown in Figure 7.4, along with the mathematical (theoretical) results computed from (7.3). As can be seen, the agreement, particularly for the radiometers covering the larger time bandwidth product, is good. The reason the results for the radiometers matched to the cells are lower than theory is probably due to the non-ideal nature of the filters. Note the 22 coefficient energy concentration filter is slightly worse than the modified sinc filter.

The results for the block detection algorithm are shown in Figure 7.5. Since our goal is to out perform a radiometer covering the entire region of interest, that theoretical curve is shown for comparison. As we see, for the particular parameters used in this simulation, the block algorithm does do somewhat better. The modified sinc filter appears to do a little better than the energy concentration filter, although the difference is slight.

**Slow FH Results.** Six sets of runs were made to simulate detection of a single cell of a Slow FH signal. Once again, the 32 coefficient modified sinc filter was used for the first three sets, with the 22 coefficient energy concentration filter used for the other three. The signal cell in this case was 800 seconds long with a bandwidth of 0.03125 Hz. (These are similar to the dimensions we used for the hopped/DS cell. For most real world spread spectrum signals, the time bandwidth product of the hopped/DS cells will be much greater than the time bandwidth product of the Slow FH cell. It was convenient here, however, to consider similar sized cells for the purposes of comparison and analysis.) In this case the amplitude was set to 0.3536 so the cell energy, once again, was 50. Each cell contained five information channels and five micro-cells.

As above, a radiometer covering the entire region of interest in the time frequency plane, and a radiometer matched to the cell's position were also simulated. These results are plotted in Figure 7.6 along with the theoretical results. Again the agreement is good. It is interesting that the results for the matched radiometer from the modified sinc filter exceed the theory while the results for the energy concentration filter fall slightly below. The difference between the effectiveness of the filters, and the difference between these results and the corresponding results in Figure 7.4 must be due to the energy distribution within the Slow FH cell. With only five micro-cells per cell, it is likely that the energy in each cell will tend to concentrate in several of the five micro-cell channels, and so slightly more energy from the sidelobes will be collected by the radiometer.

The results for the block detection scheme are shown in Figure 7.7. Once again, for the specific signal parameters used in this simulation, the algorithm outperforms the radiometer covering the entire time frequency region. The results are, as we might expect, quite comparable to those of Figure 7.5.

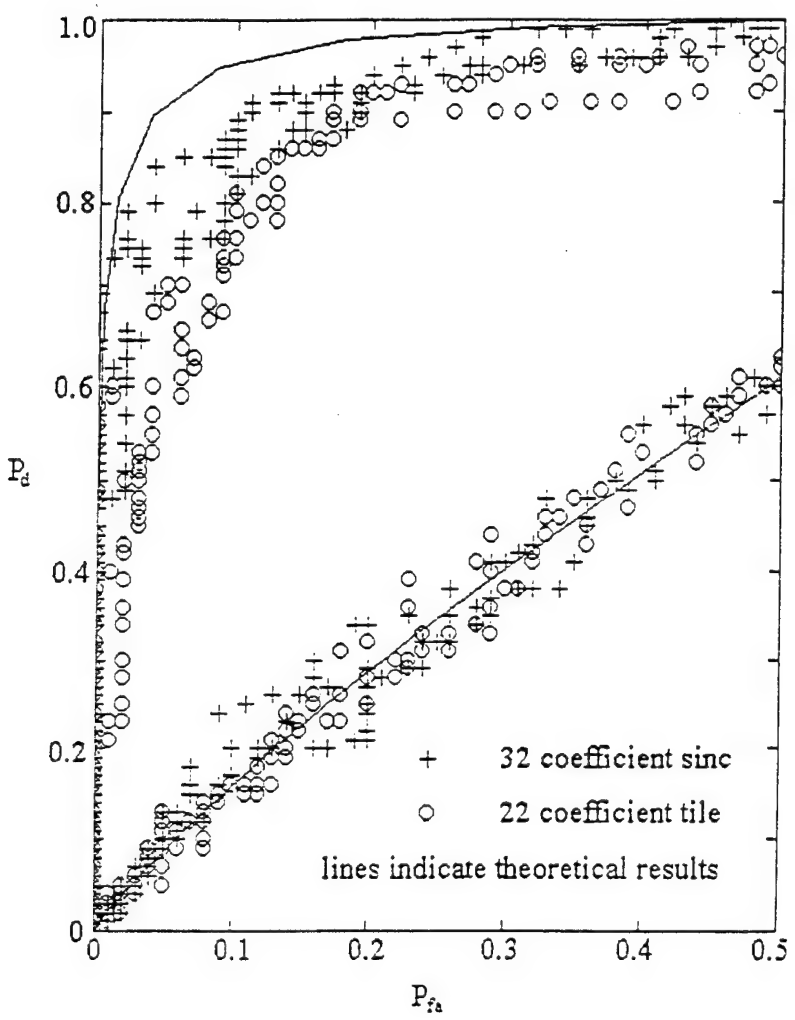



Figure 7.4. Comparison of Hopped/DS Radiometer Detection Simulation With Theoretical Predictions

**Feature Extraction**

Our goal in this section is to develop an algorithm that will estimate the cells' energy, and their positions and dimensions in the time frequency plane. Actually, using the detection algorithm, we already have a rough estimate of the cells' positions, since each cell is presumably embedded in a block whose energy exceeds the detection threshold. In the development of the feature extraction algorithm, then, we need only consider the portion of the time frequency plane covered by these blocks.

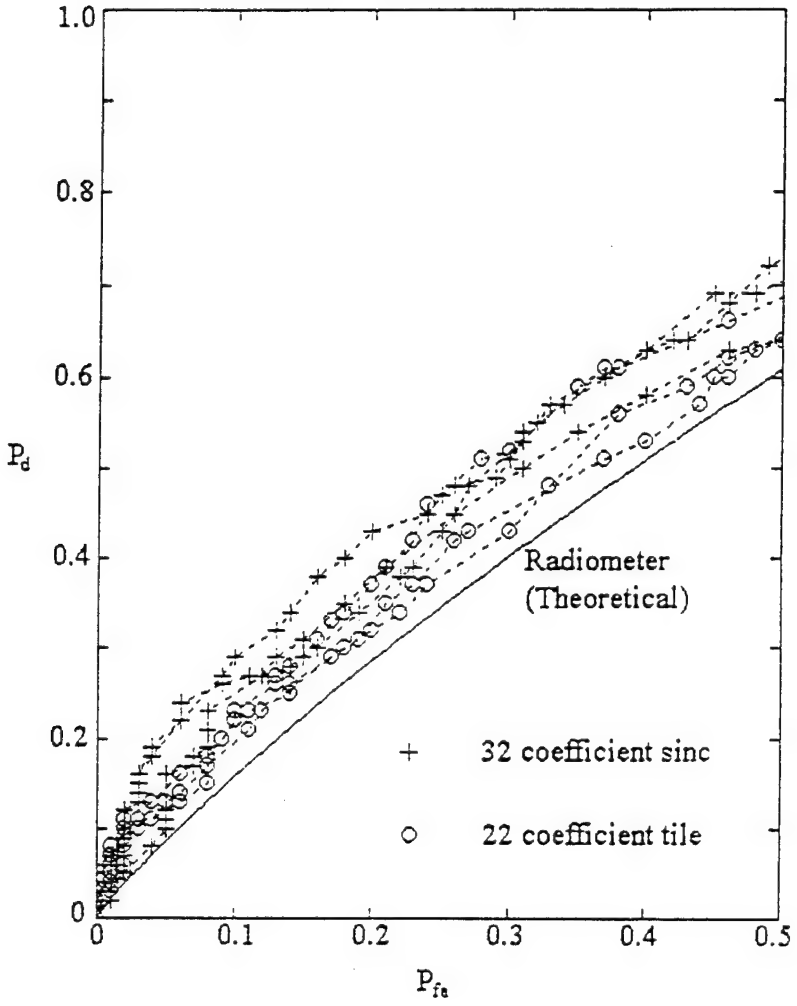



Figure 7.5. Block Algorithm Analysis Results
Detecting a Hopped/DS Signal Cell

Considering now a single block containing a cell, we have a situation somewhat similar to one described in Chapter VI, where the goal was to find a DS signal's bandwidth and center frequency. Here we want to estimate (among other things) the cell's bandwidth and center frequency, and the idea of finding the spectral vector for the block and then convolving that with various sized rectangles seems reasonable: By maximizing a test statistic similar to the one described in Equation (6.20), the corresponding rectangle size should be a fairly good estimate of the cell bandwidth. Furthermore, it is easy to see we have a similar situation when trying to find the cell's duration and location in time: We can find the block's "temporal vector", a vector formed by summing the energy in tiles corresponding to the same time intervals, and then convolve this with

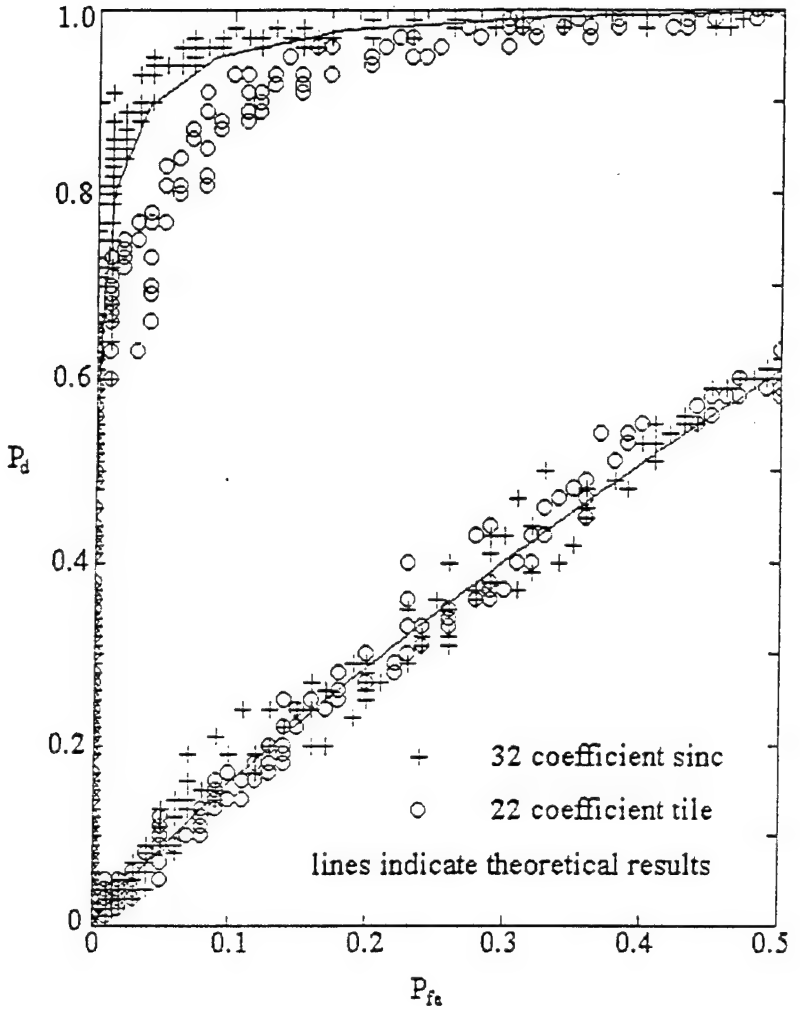



Figure 7.6. Comparison of Slow FH Radiometer Detection Simulation With Theoretical Predictions

various sized rectangles. Once again, the size of the rectangle that maximizes a test statistic can be used as an estimate of the cell duration. Naturally, we will want to use a high QMF bank tree decomposition layer to estimate the frequency parameters with high resolution, and a low layer to estimate the time parameters with high resolution.

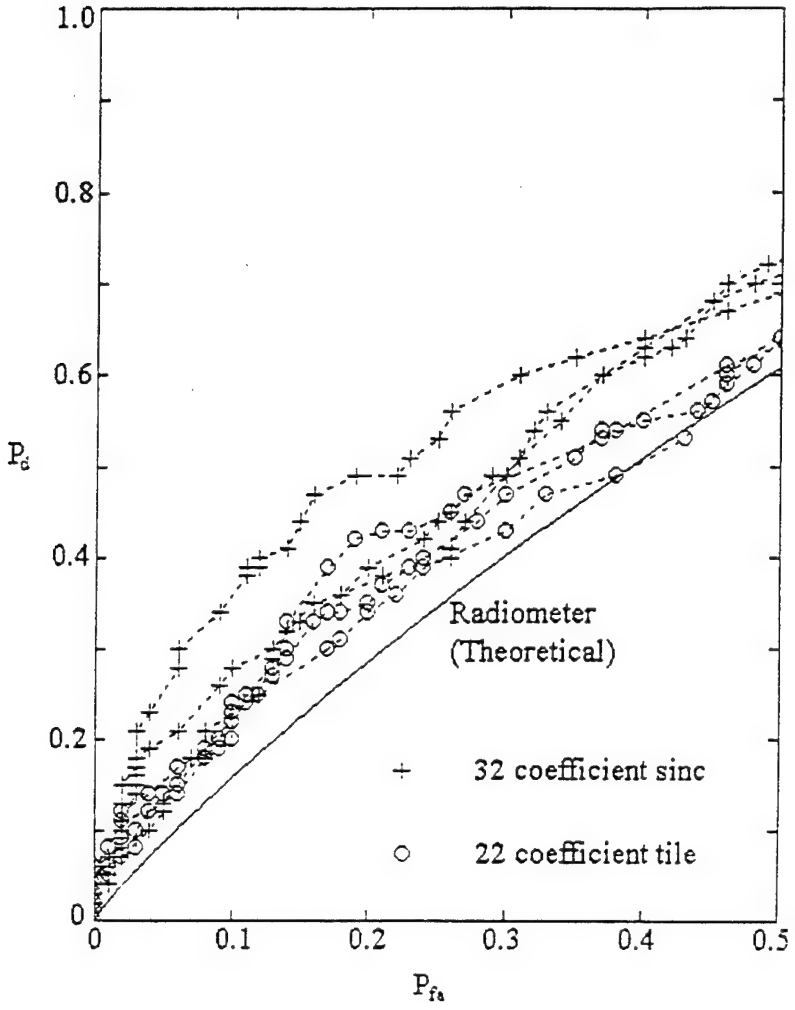



Figure 7.7. Block Algorithm Analysis Results
Detecting a Slow FH Signal Cell

The feature extraction algorithm, then, should go something like this:

1) The detection algorithm is first used to find blocks containing signal cells. As above, the particular decomposition layer used here is not critical, as long as each of the blocks' dimensions are larger than the tiles'.

2) The maximum and minimum times and frequencies are then calculated for the blocks (in seconds and hertz, as opposed to the tile indices for the particular layer).

3) A higher layer of the QMF bank tree decomposition is then used, and a spectral vector is computed. How high of a layer to use is a judgment call. The higher, the better resolution; however, tiles from higher layers have a longer duration, and will, therefore, contain more noise from regions of the time frequency plane where there is no cell energy.

4) Once a spectral vector is obtained, a "rectangle vector" is formed (as in Chapter VI) with a width equal to the minimum bandwidth cell that the interceptor is interested in finding. The rectangle vector is then convolved with the spectral vector, the position of the maximum value of the resulting vector is noted, and a test statistic (described below) is found. The rectangle vector's size is then increased by one, and the step is repeated until the rectangle equals the maximum sized cell that the interceptor is interested in finding.

5) The test statistics for the various sized rectangles are then compared, the maximum is found, and the corresponding rectangle's size and position are taken to be estimates of the cell's bandwidth and location (in frequency).

A similar process is used to find the time estimates. The trade-offs in choosing a good low layer of the QMF bank tree decomposition are analogous to those discussed for the high layer.

As discussed in Chapter VI, we cannot compare the energy found between convolutions of the spectral vector and different sized rectangles, because, as the rectangles become larger, the amount of both signal energy and noise energy is increased. Of course, a similar situation will occur here with the temporal vector. We solved this in Chapter VI by finding Equation (6.20), a test statistic that is adjusted for the rectangle's size and that, when maximized, maximizes the probability of detection of the signal. Here, we need a similar test statistic, except, rather than adjusting it just for the spectral rectangle's size, we also want to adjust it for the temporal rectangle's size. With the same logic that we used to derive (6.20), we can begin with (7.3) and come up with

$$u = \frac{\varepsilon_o - N_0 T_r W_r}{\sqrt{T_r W_r}} \tag{7.7}$$

where

- $\varepsilon_o$ is the observed energy (the high value resulting from the convolution)
- $N_0$ is the one sided noise energy density
- $T_r$ is the size of the rectangle when convolving with the temporal vector; the block's duration when convolving with the spectral vector
- $W_r$ is the size of the rectangle when convolving with the spectral vector; the block's bandwidth when convolving with the temporal vector

As a variation of the algorithm, either the time or frequency parameters could be found using the entire region of the block, as described above, and then those results could be used to reduce the region of the block used to find the other (frequency or time) parameters. Although this would tend to increase the amount of signal energy versus noise energy used in the second set of estimates, any errors in the first would hurt the second. (We do not explore this further here.)

The cell's energy can either be estimated using the block's energy (subtracting the expected noise energy), or by using the cell position and dimension estimates to find the energy in that particular region of the time frequency plane. The problem with the second possibility is that errors in the position and dimension estimates will increase the error in the energy estimate.

Analysis

The purpose of this analysis was to verify the ability of the feature extraction algorithm to estimate the signal parameters of interest, and to determine how well it works with real (as opposed to ideal) QMF filters. This was done via simulation. These were set up similar to the detection simulations whose results are described above. A single hop cell was randomly positioned on the time frequency plane along with WGN. The cell parameters were identical to those used in the detection simulations, except for the cell energy which was increased by ten times to $\varepsilon = 500$. The scenario is that the detection of the signal by the interceptor is practically a certainty, and the goal now is to determine the features. For each of the two types of hop cells, two sets of runs were made; one each for the 32 coefficient modified sinc filter and the 22 coefficient energy concentration filter.

**Hopped/DS Results.** The error in the time position for each run was computed by first averaging the estimated beginning and end time for the cell, and then subtracting the known center time. The results are shown in Figure 7.8. The mean values of the results are -3.58 seconds for the modified sinc filter and -3.42 seconds for the energy concentration filter.

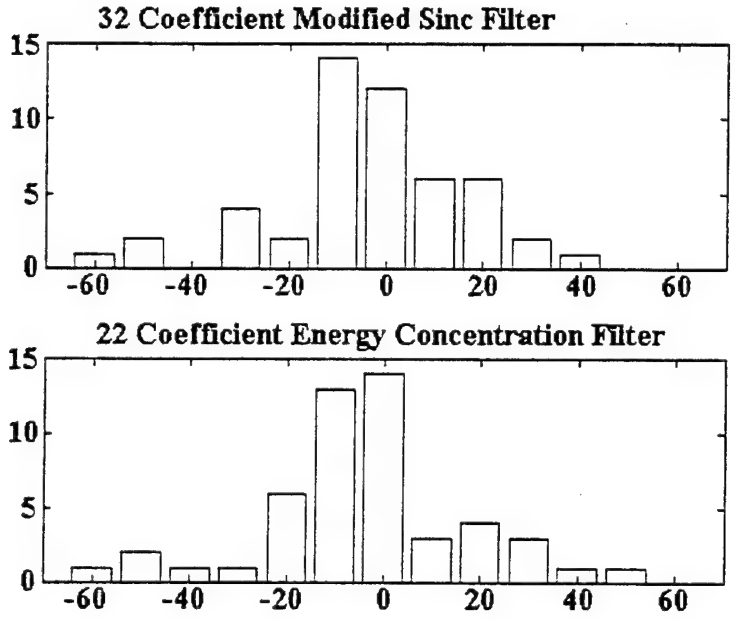



Figure 7.8. Error in Estimates of Time Position of Hopped/DS Cells
(Horizontal Axis is Seconds, Vertical is the Number of Estimates in Bin)

The error in the frequency position for each run was computed by first averaging the estimated lower and upper frequencies for the cell, and then subtracting the known center frequency. The results are shown in Figure 7.9. The mean values of the results are $-0.173 \times 10^{-3}$ Hz for the modified sinc filter and $-0.075 \times 10^{-3}$ Hz for the energy concentration filter.

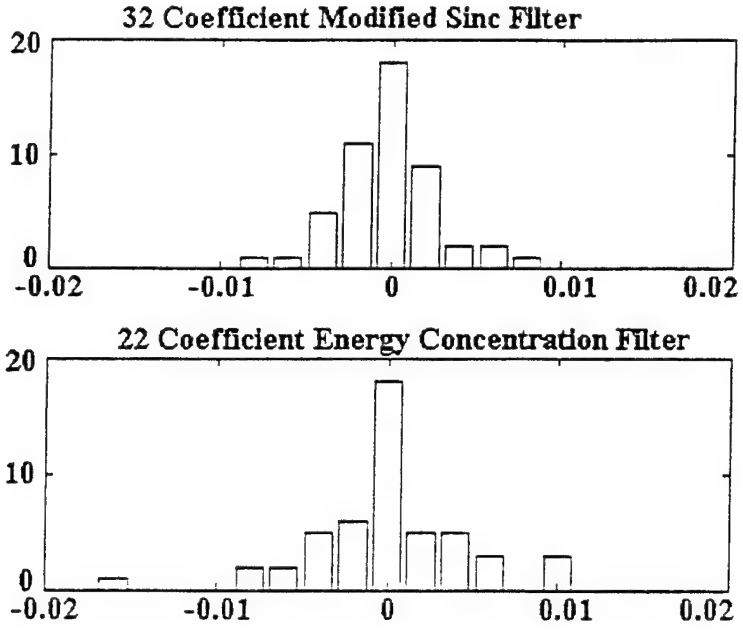



Figure 7.9. Error in Estimates of Frequency Position of Hopped/DS Cells
(Horizontal Axis is Hertz, Vertical is the Number of Estimates in Bin)

The estimates of the duration of the cells were made simply by subtracting the beginning time estimate from the end. The results are shown in Figure 7.10, and the mean values are 788 seconds and 791 seconds for the modified sinc filter and energy concentration filter respectively. Since the true cell duration is 818 seconds, it appears the energy at very beginning and end of the cells are dispersed outside of the block. To see why this may be, consider the beginning of the cell: The transition, being instantaneous, contains frequency components greatly exceeding the Nyquist rate and, therefore, aliased across the band. The same effect, of course, would occur as the cell is turned off. It seems likely, then, that the algorithm's estimate of cell duration will always have a slight bias.

The estimate of the cells' bandwidths were made by subtracting the upper frequency estimate from the lower, and the results are shown in Figure 7.11. The mean values are 0.0295 Hz and 0.0333 Hz for the modified sinc filter and energy concentration filter respectively. The true cell bandwidth is 0.0305 Hz and so the means are pretty close, however, as can be seen in the figure, the variation in estimates for individual cells is large.

The energy estimates made by finding the layer seven tiles that encompass the estimated cell positions and adding those tiles' energies are shown in Figure 7.12. This includes both the cell energy and noise energy, and so the expected observed energy is

$$E[\varepsilon_o] = E[\varepsilon_c] + TBN_0 = 500(0.774) + 50 = 437 \tag{7.8}$$

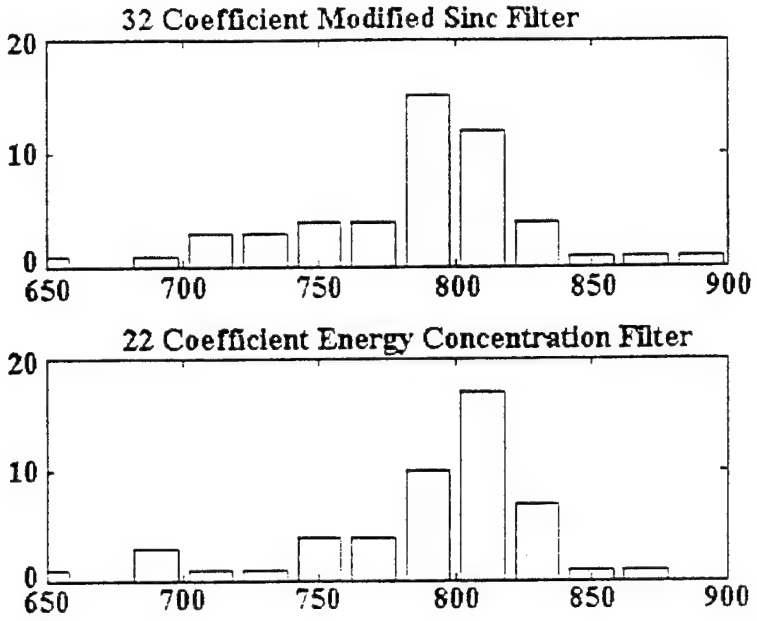



Figure 7.10. Estimates of Time Duration of Hopped/DS Cells
(Horizontal Axis is Seconds, Vertical is the Number of Estimates in Bin)

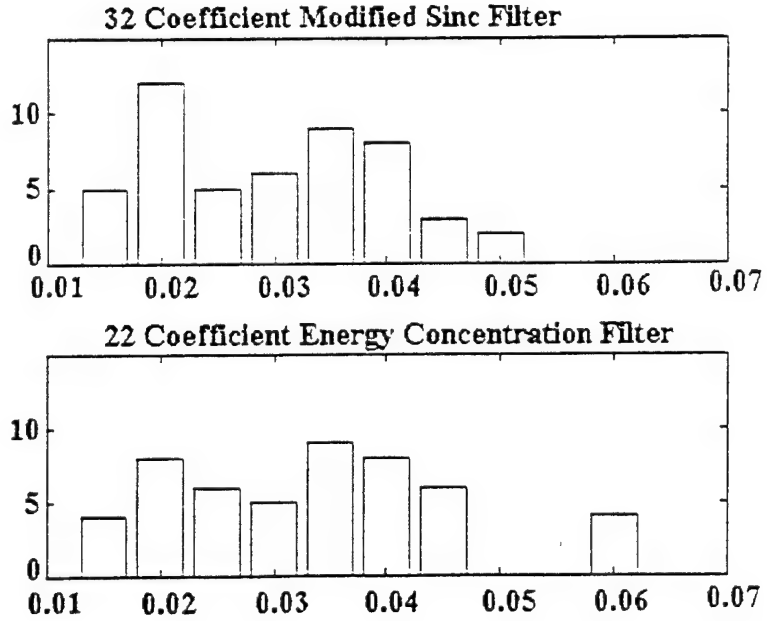



Figure 7.11. Estimates of Bandwidth of Hopped/DS Cells
(Horizontal Axis is Hertz, Vertical is the Number of Estimates in Bin)

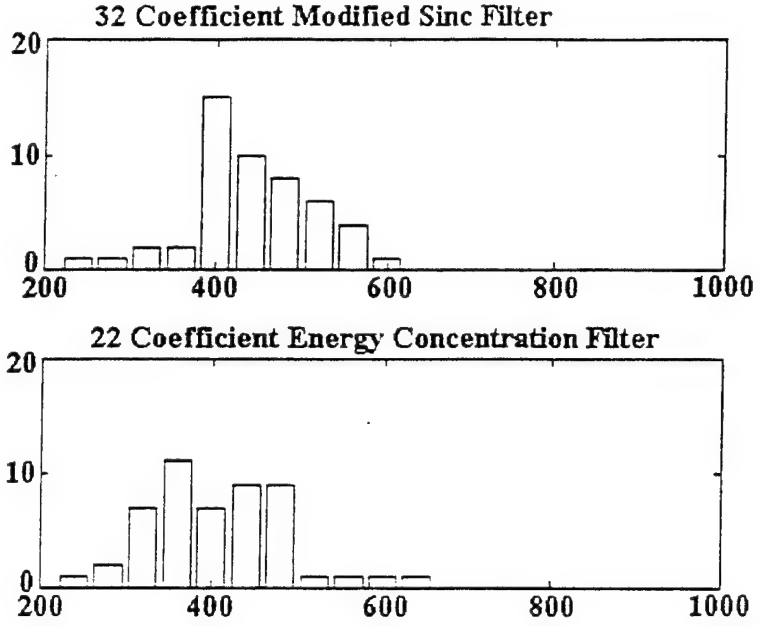



Figure 7.12. Estimates of Energy of Hopped/DS Cells Plus Noise
(Horizontal Axis is Energy, Vertical is the Number of Estimates in Bin)

where $\varepsilon_c$ is the cell energy collected, and we expect to collect 0.774 of the total cell energy within the bandwidth of the cell (see Figure 6.4). The cell time bandwidth product is 25, which, with a single sided noise energy density of two, leads to the result for the second term. The mean energy collected with the modified sinc filter is 443, a value close to the one expected. The mean energy collected with the energy concentration filter is slightly less, 408.

The block energy collected is shown in Figure 7.13. The expected value is

(7.9) $$E[\varepsilon_o] = E[\varepsilon_c] + t_0 W N_0 = 500(0.903) + 200 = 652$$

Here, we assume the block is approximately centered over the cell and, since its bandwidth is twice the cell bandwidth, we collect 0.903 of the cell energy. The block's time bandwidth product is 100 leading to the result. The actual mean found using the modified sinc filter is 670, while the mean found using the energy concentration filter is 614.

**Slow FH Results.** The error in time position for each run was computed in the same manner as described for the hopped/DS simulation. The results are shown in Figure 7.14. The mean for the modified sinc filter is -3.54 seconds, while the mean for the energy concentration filter is -3.06 seconds.

The error in frequency position was also computed in a manner similar to that described for the hopped/DS simulation, and the results are shown in Figure 7.15. Here, the mean for the modified sinc filter is -0.532 Hz, while the mean for the energy concentration filter is -0.298 Hz.

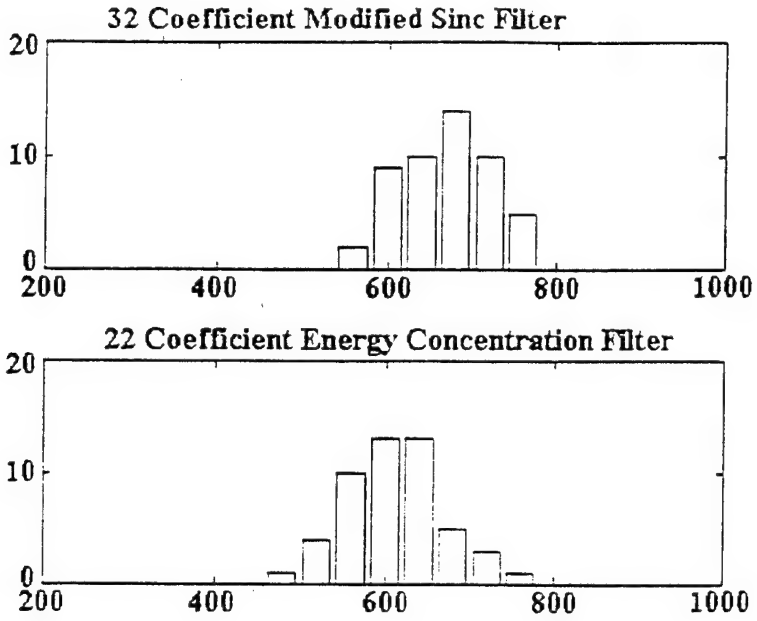



Figure 7.13. Energy of Blocks That Include a Hopped/DS Cell Plus Noise
(Horizontal Axis is Energy, Vertical is the Number of Estimates in Bin)

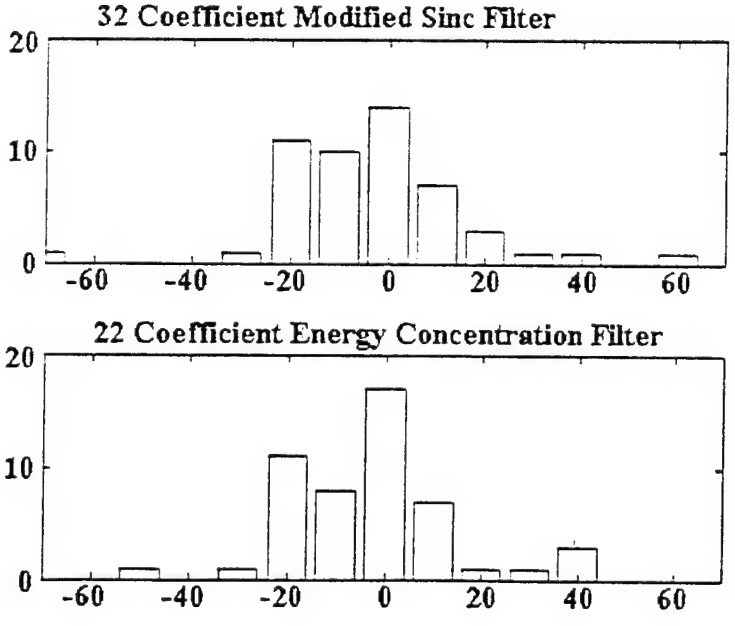



Figure 7.14. Error in Estimates of Time Position of Slow FH Cells
(Horizontal Axis is Seconds, Vertical is the Number of Estimates in Bin)

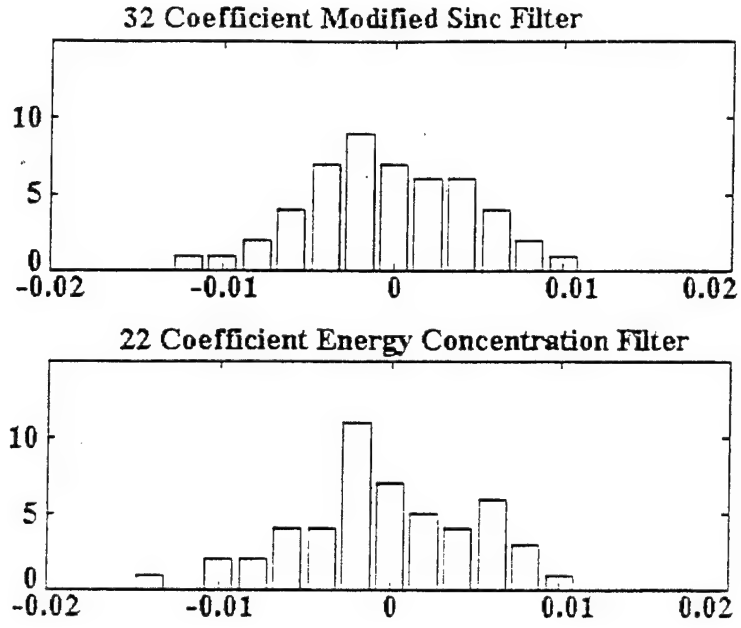



Figure 7.15. Error in Estimates of Frequency Position of Slow FH Cells
(Horizontal Axis is Hertz, Vertical is the Number of Estimates in Bin)

The estimates of cell duration are shown in Figure 7.16. The mean for the modified sinc filter is 774 seconds, while the mean for the energy concentration filter is 780 seconds. Here, as with the hopped/DS cells, we should expect to see a slight bias resulting from the spread of energy due to the instantaneous turning on and off of the cells. Since the Slow FH cells used in the simulation are 800 seconds long, a bias is indeed evident.

The estimates of cell bandwidth are shown in Figure 7.17. The mean for the modified sinc filter is 0.023 Hz, while the mean for the energy concentration filter is 0.024 Hz. Since the true cell bandwidth is 0.03125 Hz, these estimates appear not to be very good. What is apparently happening, is that, because there are five information channels and only five micro-cells in each cell, many cells have energy concentrated within a few channels.

The energy estimates made, as for the hopped/DS simulations, by finding the layer seven tiles that encompass the estimated cell positions, and adding those tiles' energies, are shown in Figure 7.18. This includes both the cell energy and noise energy, and so the expected observed energy is

$$E[\varepsilon_o] = E[\varepsilon_c] + TBN_0 = 500 + 50 = 550 \tag{7.8}$$

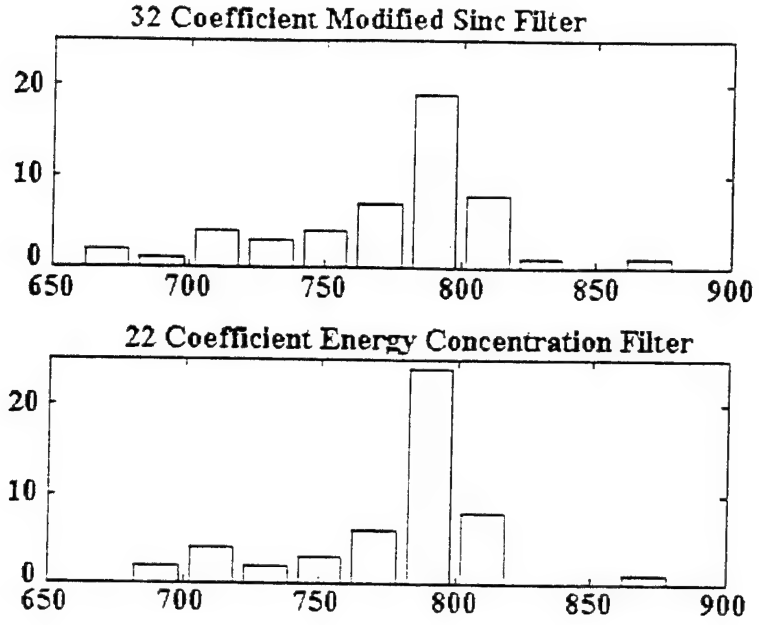



Figure 7.16. Estimates of Time Duration of Slow FH Cells
(Horizontal Axis is Seconds, Vertical is the Number of Estimates in Bin)

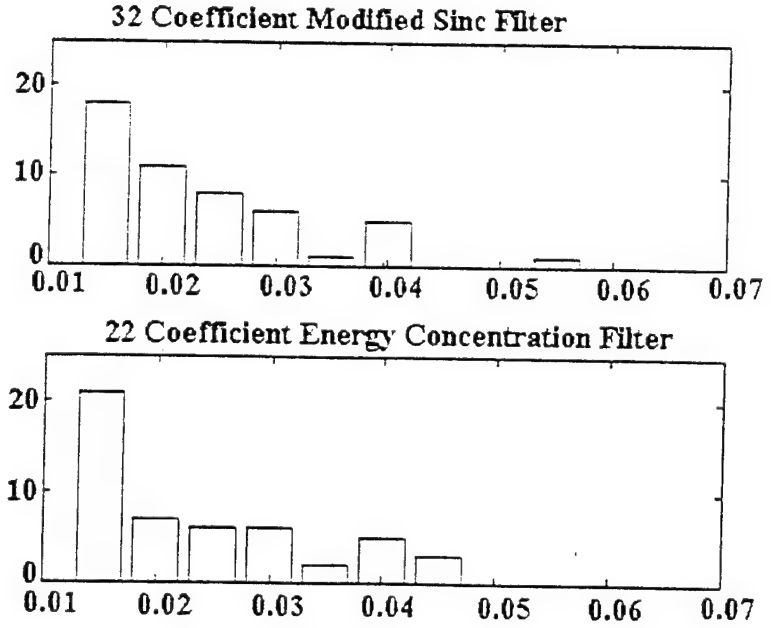



Figure 7.17. Estimates of Bandwidth of Slow FH Cells
(Horizontal Axis is Hertz, Vertical is the Number of Estimates in Bin)

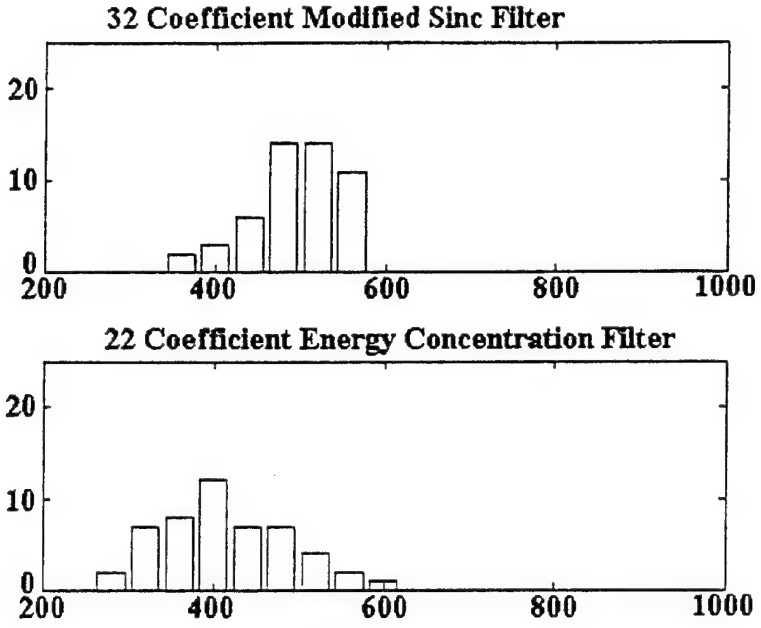



Figure 7.18. Estimates of Energy of Slow FH Cells Plus Noise
(Horizontal Axis is Energy, Vertical is the Number of Estimates in Bin)

where, due to the energy distribution of the Slow FH signal cells, as described in Figure 7.1, we expect to collect essentially all of the cells' energy. As with the hopped/DS simulations, the cell time bandwidth product is 25. The mean energy collected with the modified sinc filter is 492, while the mean energy collected with the energy concentration filter is 416. These low estimates may be due to the error in estimating the cells' dimensions.

The block energy collected is shown in Figure 7.19. The expected value is

$$E[\varepsilon_o] = E[\varepsilon_c] + t_0 W N_0 = 500 + 200 = 700 \tag{7.9}$$

The actual mean using the modified sinc filter is 721, while the mean found using the energy concentration filter is 660.

## Summary

We began this chapter by describing the signals we are interested in intercepting: The hopped/DS signals, which are hopped in frequency, time, or both, and also spread with a random binary waveform; and Slow FH signals, which are hopped, and spread by the FSK coding of the signal's information.

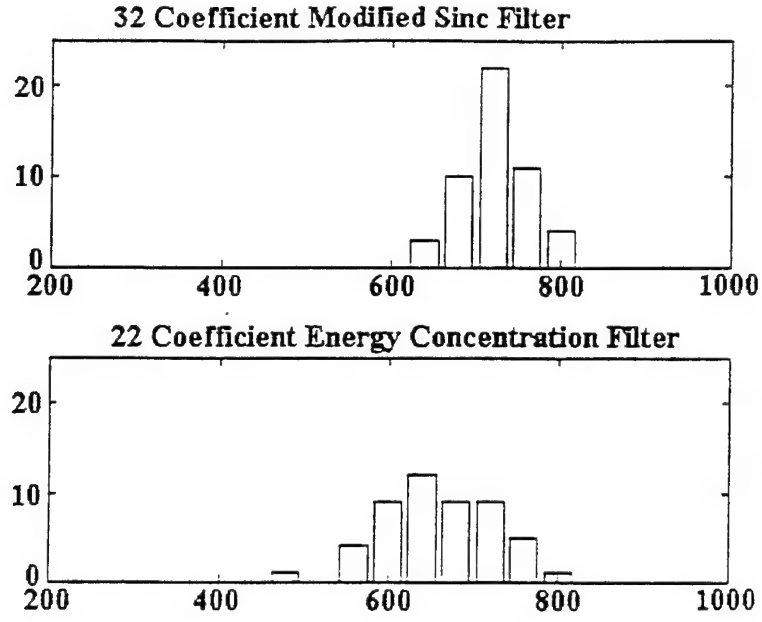



Figure 7.19. Energy of Blocks That Include a Slow FH Cell Plus Noise
(Horizontal Axis is Energy, Vertical is the Number of Estimates in Bin)

We then looked at detection, and decided the best method, given the interceptor's knowledge of the signal structure, is to use a block algorithm, similar to that described in Chapter V, except with larger blocks. In fact we decided that, given the interceptor's limited knowledge about the range of cell dimensions, a scheme in which the blocks equal or exceed the maximum possible size of the cells in each dimension is best. Simulations of this scheme show that a performance better than that of a simple radiometer is achievable under some conditions. Further investigation in this area is, of course, possible. In particular, one might try detecting signals in which more than one cell is present during the observation interval. In this case, rather than simply comparing the highest energy block to a threshold, the probability of false alarm may be decreased by requiring a certain number of blocks to exceed a threshold in order to declare a signal to be present.

Finally, we looked at a method for finding estimates of the cells' dimensions and locations in the time frequency plane, and also estimates of the energy in each cell. It appears that most of the estimates are fairly good, except perhaps for the estimates of the bandwidth. In the next chapter, we will use some of these estimates in an effort to find characteristics to distinguish between hopped/DS and Slow FH cells.

## VIII. Characterizing Energy Cells by Their Frequency Distribution

### Introduction

In Chapters V-VII, we found algorithms that estimated the dimensions of the DS signal, or the hopped signals' cells, in the time frequency plane. Even though we developed the algorithms mathematically using specific distributions of cell energy (usually under a rectangular envelope in the time dimension and, therefore, sinc squared in the frequency dimension), we didn't fully exploit this knowledge. All we were really interested in was the fact that most of the energy was concentrated in a small region of the plane. This is reflected in our feature extraction algorithms in Chapters VI and VII, where we used rectangular windows, rather than windows whose shape is more closely matched to the signal's characteristics.

One advantage of this is that we should have generally good performance for a wide variety of signal energy distributions. If the transmitter did not, as we assumed, transmit at full power for the duration of a hop cell, but instead varied the amplitude over the duration of the cell, under a Gaussian shaped envelope for example, our interceptor would still be capable of making a good estimate of the cell's dimensions. Although these numbers would have a slightly different meaning than for the rectangle or sinc squared functions, they would be consistent from cell to cell, and would therefore allow a classifier, in an intercept receiver architecture like that of Figure 1.2, to determine whether the relevant cells may have come from a single transmitter.

In this chapter we will do something different, and look at how an interceptor can exploit the knowledge he might have of the energy distribution from a particular type of spread spectrum transmitter to decide whether a detected cell came from that type of transmitter. Specifically, the interceptor will assume, as a null hypothesis, that a detected cell has a particular distribution of energy in frequency, and we will describe an algorithm in which this hypothesis is tested. After describing the algorithm, simulation results will be presented to verify it works. In our specific case, we will use the sinc squared energy distribution of the hopped/DS signal cell as the null hypothesis, and the Slow FH cell as the alternative.

Finally, as we will see, the algorithm suggests a method for estimating the center frequency and bandwidth for the hopped/DS cells that may be better than the one described in Chapter VII. This will be discussed, the relationship between these methods will be explored, and simulation results will be presented for comparison with those of the previous chapter.

### Distinguishing Between the Hopped/DS and Slow FH Cells

For this scenario, we will assume the interceptor has used the block algorithm discussed in Chapter VII and has detected a hop cell with a time bandwidth product larger than one. The question then is: Is there some way for the interceptor to distinguish between the hopped/DS cell and the Slow FH cell based on the distribution of energy within the block?

It should probably be noted that this scenario is more of a mathematical exercise than real life problem. In a practical situation, the time bandwidth product of the hopped/DS cell will generally be much greater than that for the Slow FH signal. In that case, a better problem might be to see if the interceptor can distinguish between a hopped/DS cell like the one used in Chapter VII, and any other one whose amplitude is varied by the transmitter over the duration of the cell. The algorithm discussed here, however, should be applicable to that case also.

We will use the characteristics of the block's spectral vector to conduct the test. Initially one might think that some sort of algorithm to look for the energy concentrations due to the micro-cells in the Slow FH cell may be more effective. The problem with this approach is that the micro-cells, being closely spaced in the time frequency plane, are difficult to resolve.

For our purposes, as a demonstration of concept exercise, the Slow FH cell, because of the wide variation in spectral characteristics due to the positioning of the micro-cells, is ideal. Rather than trying to distinguish between two known distributions, the interceptor is faced with the more difficult problem of distinguishing a cell with the known distribution from any other. Our test then, in terms of the null hypothesis, $H_0$, and the alternative, $H_a$, is

(8.1)
$H_0$: Block's Spectral Vector Has Sinc Squared Distribution + Noise
$H_a$: Block's Spectral Vector Has Other Distribution + Noise

Tests similar to this are widely discussed in the literature. For example it is similar in many ways to the Chi-Squared test to determine if a set of statistical values were drawn from a given probability distribution function (pdf) [14] [29] [40]. For our test, the strategy will be simply to form a spectral vector from the block containing the signal cell (as described in Chapters VI and VII), decide how much energy we expect to see in each bin, and then find the square of the difference between that and the amount actually found. Our test statistic will be the sum of these values.

Given a block containing a hopped/DS cell, and considering for the moment a case when the cell is located in the center of the block and has a known bandwidth, we should expect the energy to be distributed as shown in Figure 8.1. The expected noise energy in each bin would be

(8.2) $$E[N_b] = N_0 t_o W_b$$

where

| | |
|---|---|
| $N_b$ | is the noise energy in each bin |
| $N_0$ | is the one sided noise density (usually normalized to two) |
| $t_o$ | is the block's duration |
| $W_b$ | is each bin's bandwidth |

while the expected signal energy would be approximately

(8.3) $$E[\varepsilon_k] = \varepsilon \frac{W_b}{B} \mathrm{sinc}^2\left(\frac{f_k - f_c}{B}\right)$$

where

| | |
|---|---|
| $\varepsilon_k$ | is the signal energy in the k-th bin |
| $\varepsilon$ | is the total cell energy |
| B | is the hopped/DS cell's bandwidth (as defined in Figure 6.4) |
| $f_c$ | is the hopped/DS cell's center frequency |
| $f_k$ | is the k-th bin's center frequency |

Here, we are assuming the bin bandwidth is small enough relative to the cell bandwidth that we can approximate the area under the sinc squared function in each bin with a rectangle whose height is the value of the function at the center of the bin.

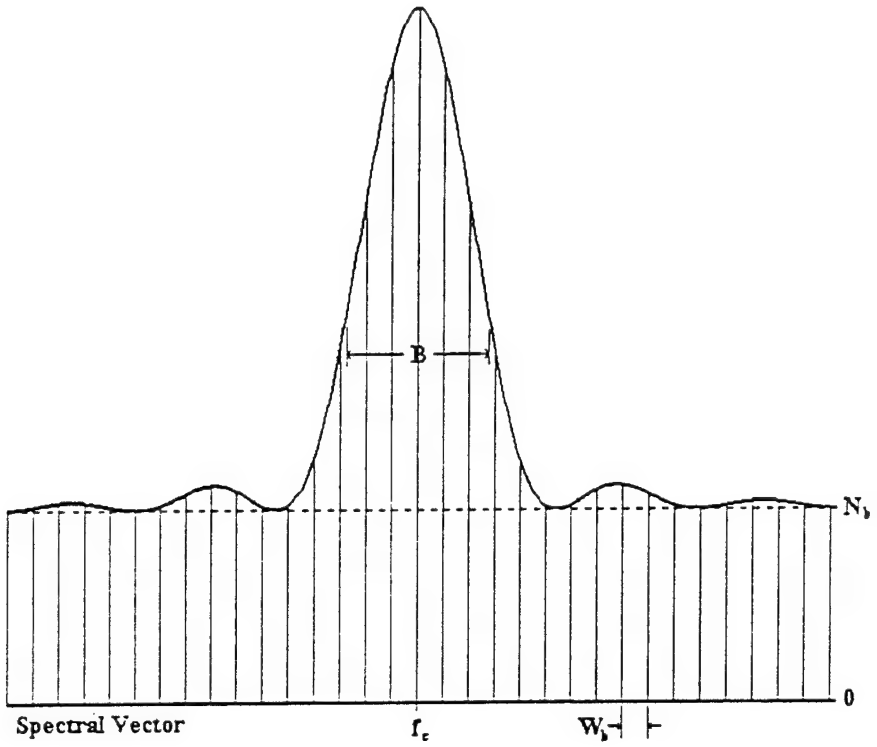



Figure 8.1. Expected Energy Distribution in a Block's Spectral Vector
When a Hopped/DS Cell is Present and Centered

Now, if we further assume, for the moment, that the signal energy distribution is deterministic, so $\varepsilon_k = E[\varepsilon_k]$, we will have the case where the pdf of total energy in each bin is the function in (2.29): Chi-Squared with $t_o W_b$ degrees of freedom, and with a non-centrality parameter equal to $\varepsilon_k$. For a reasonably large value of $t_o W_b$, we can, as we did in Chapters VI and VII, approximate this with the Gaussian distribution with a mean of $\varepsilon_k + N_0 t_o W_b$ and variance of $2N_0\varepsilon_k + N_0^2 t_o W_b$. From this, we can form

$$y_k = \frac{v_k - \varepsilon_k - N_0 t_o W_b}{\sqrt{2N_0\varepsilon_k + N_0^2 t_o W_b}} \tag{8.4}$$

where $v_k$ is the observed energy in the k-th bin. If our assumptions were valid, the pdf from which $y_k$ was drawn would have a Gaussian distribution with zero mean and variance of one. This is a

measure of the difference between the expected energy and true energy in each bin. We can combine these for all of the K bins in the spectral vector to form the test statistic

$$z = \sum_{k=1}^{K} (y_k)^2 \tag{8.5}$$

where we square each value of $y_k$ to prevent positive and negative values from canceling. Again if the assumptions were valid, the distribution of this statistic would be Chi-Squared, with K degrees of freedom. The test, then, would be similar to the Chi-Squared test of a pdf: If the cell really is hopped/DS, the value of z should be small, and a threshold could be set based on K and the acceptable probability of a Type I error (in this case, deciding the cell is not hopped/DS, when it actually is).

As it happens, however, the signal energy is not deterministic. The sinc squared function is the Fourier Transform of an infinitely long random binary sequence spreading the signal. Since, however, we actually have a relatively short duration cell, the length of the random binary sequence is so short that the true energy distribution will be a bit different in every case, and the sinc squared function is better regarded as an average distribution of the signal energy in the frequency dimension. We can overcome this difficulty by using the expected signal energy for each bin from (8.3) in place of $\varepsilon_k$ in (8.4). This non-deterministic nature of the signal energy will cause a greater variance on the test statistic z than predicted from the analysis above. If we use hard thresholds (as we do in the simulations described below) the practical result is that we will have to increase them to achieve the same probability of avoiding a Type I error.

It may be possible to carry the mathematical analysis further and determine a pdf for $\varepsilon_k$. However, we will not explore this further here. Our goal is merely to demonstrate that the distribution of the spectral vector is a credible method for estimating the character of the signal cell. Also, any analysis would depend on the specific spreading technique used by the transmitter. Although we are assuming the signal is spread with a random binary waveform, a more general application of the techniques described in this chapter may face a Hopped/DS cell spread with some other random (to the interceptor) waveform. In an actual receiver architecture such as the one in Figure 1.2, the determination of cell energy distribution would not be made using single cells in each observation interval in conjunction with a hard threshold, but rather would be made with an ensemble of z values for many cells and an adaptive classifier. In that case, the values for z would be passed, along with the other estimated cell information, in the list from the algorithm block to the classifier block.

So far we have only considered the case where the interceptor knows both the center frequency and bandwidth of the cell. When the center frequency is unknown, it is a reasonable adaptation of the algorithm above to form a vector of expected energy, and then test it by trying every possible alignment. This means taking the vector of expected values and "shifting" it along the spectral vector, much as in the convolution operation. In this case, however, we take the difference in energies for each bin and form z, using the lowest value found as the test statistic. Notice, this also gives us an alternative method, from that described in Chapter VII, for estimating the cell's center frequency.

When the interceptor knows neither the center frequency nor bandwidth of the cell, a further adaptation can be made by trying different values for B when computing the vector of expected energy values. As different values are tried, and each is shifted across the spectral vector, values of z are computed and, it is easy to see, the smallest should correspond to the closest alignment of center

frequencies and bandwidths. (Giving us an alternative method, from that in Chapter VII, of estimating the hopped/DS cell bandwidth.)

Incorporating these changes in the algorithm into (8.5), we have

$$z = \min_{m,B} \left\{ \sum_{k} \frac{\left[v_k - g_{k+m}(B)\right]^2}{2N_0 E\left[\varepsilon_{k+m}(B)\right] + N_0 N_b} \right\} \tag{8.6}$$

where m is the offset as the vectors of expected and observed energy are shifted relative to each other. (The vectors should be padded on either end by zeros, as necessary.) $g_{k+m}$ is the vector of expected energy, and is

$$g_{k+m}(B) = E\left[\varepsilon_{k+m}(B)\right] + N_0 t_o W_b \tag{8.7}$$

where $E\left[\varepsilon_{k+m}(B)\right]$ is the expected signal energy, for a hopped/DS signal with bandwidth B, in bin k+m found using (8.3). (Notice the expected energy in the k+m th bin is also used in the denominator of (8.6).)

In general, the actual values of $f_c$ and B, and the tested values will not be exactly matched, since we can't expect to have the cell center frequency align with the spectral vector bins (which correspond to the QMF bank tree tiles), and we can only try a discrete number of bandwidths. The first problem can be reduced by examining spectral vectors formed from higher layers of the QMF bank tree (yielding narrower bin bandwidths). The second can be reduced by using small increments between tested bandwidths. In the simulations whose results are described below, the increments used corresponded to the bin widths of the layer examined. Using high layers of the QMF tree also reduces the error in (8.3). In fact, the reason for using (8.3) in our algorithm, rather than the more accurate (but more computationally intense) estimate in which the integral of the sinc squared function over the width of the bin is computed, is that the error is only of the order of that due to the mis-match in center frequency and bandwidth estimates.

To summarize the algorithm:

1) Use the block algorithm described in Chapter VII to find blocks containing signal cells.

2) Select the values of bandwidth to test.

3) Form a vector of expected energy values using (8.7)

4) Shift the vectors and test for each value of m, then go to the next value of B and repeat 3), to find the minimum test statistic as specified in (8.6).

5) Pass the resulting z, for each block, to the classifier block. Or, for our simulations, test z against a threshold and, if z is lower, decide the cell is hopped/DS; otherwise, decide it is not.

**Analysis Results**

A simulation analysis of this algorithm was carried out using code described and listed in Appendix E. First, 100 runs were made, each with a waveform consisting of WGN and a single cell of a hopped/DS signal. The cell parameters were identical to the ones used in the simulations in Chapter VII: The duration was 818 seconds (normalized), the bandwidth was 0.0305 Hz, to yield a time bandwidth product of 25, and the cell energy was 500 (with a normalized one sided noise energy density of one). The cell was found using the block algorithm described in Chapter VII and the test statistic was computed using the algorithm described above with the spectral vector formed from the layer seven output of the QMF bank tree. This was tested against a vector of hard thresholds and the probability of making a Type I error (deciding the cell is not a hopped/DS cell) was found.

Then, 100 runs were made with a waveform consisting of WGN and a Slow FH signal. Again the cell parameters were the same as those used in Chapter VII: The duration was 800 seconds and the bandwidth was 0.03125 Hz, so the time bandwidth product was also 25. As above, a cell energy of 500 was used. The test vector was computed using the algorithm described above, and this was tested against the same vector of hard thresholds used for the hopped/DS cells. This yielded a probability of correctly deciding the cell was not hopped/DS.

As in previous chapters, these simulations were repeated three times each for the 32 coefficient modified sinc filter and the 22 coefficient energy concentration filter. The results are presented in Figure 8.2. The vertical axis, in the figure, represents the probability of the interceptor correctly rejecting the null hypothesis (deciding the cell is not hopped/DS) when the actual cell is Slow FH. The horizontal axis represents the probability of the interceptor making a Type I error: Incorrectly rejecting the null hypothesis when the cell is, in fact, hopped/DS. As can be seen, both sets of filters yield a similar performance.

As discussed above, this algorithm can also be used as an alternative to the feature extraction algorithm in Chapter VII for estimating the center frequency and bandwidth of the hopped/DS cell. Because the algorithm in this chapter uses more information: The expected shape of the spectral vector, rather than simply the greatest concentration of energy, there is reason to expect it may work better.

The code listed in Appendix E was also used to make these estimates. To make the comparisons with those in Chapter VII direct, fifty runs were made, each with a hopped/DS cell with the parameters as described above. The spectral vector was formed using the layer ten output from the QMF bank tree.

The error in the estimate of the center frequencies is shown in Figure 8.3, and can be compared to the error in the Chapter VII algorithm shown in Figure 7.9. The histograms show no clear benefit of one algorithm over the other.

The bandwidth estimates are shown in Figure 8.4, and can be compared to the Chapter VII algorithm's estimates shown in Figure 7.11. For these estimates, the algorithm used in this chapter is clearly superior.

The experiment was also conducted using Slow FH signal cells, but, not surprisingly, the estimates of center frequency and bandwidth were no better than those made with the algorithm in Chapter VII.

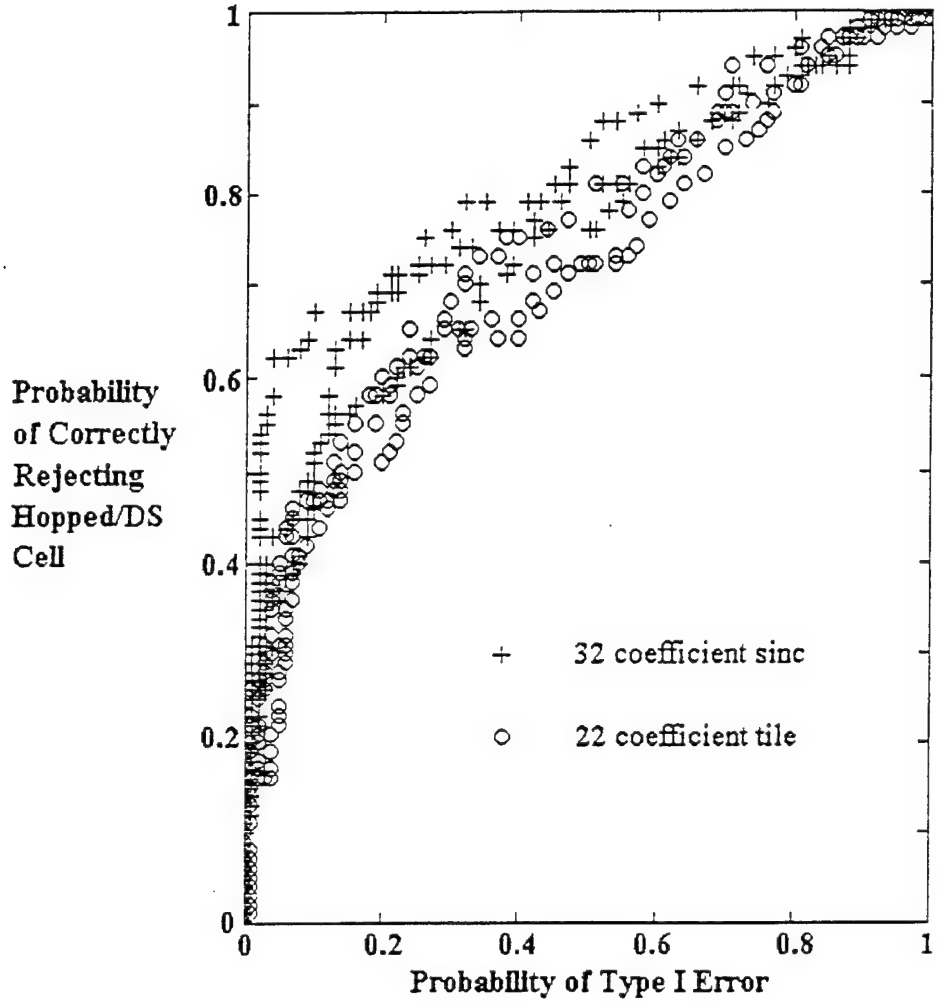



Figure 8.2. Results of Test to Determine Whether Cell is Hopped/DS

### Summary and Conclusions

This chapter very briefly explored the use of a least squares algorithm to distinguish between cells by the distribution of their energy in frequency. This is nothing revolutionary, and we did not go into very great depth. Rather, we merely went far enough to illustrate that the concept will work for an intercept receiver with an architecture like that in Figure 1.2. In addition, we saw that it offers a good method for estimating the bandwidth of the cell, when the general shape of the distribution is known to the interceptor.

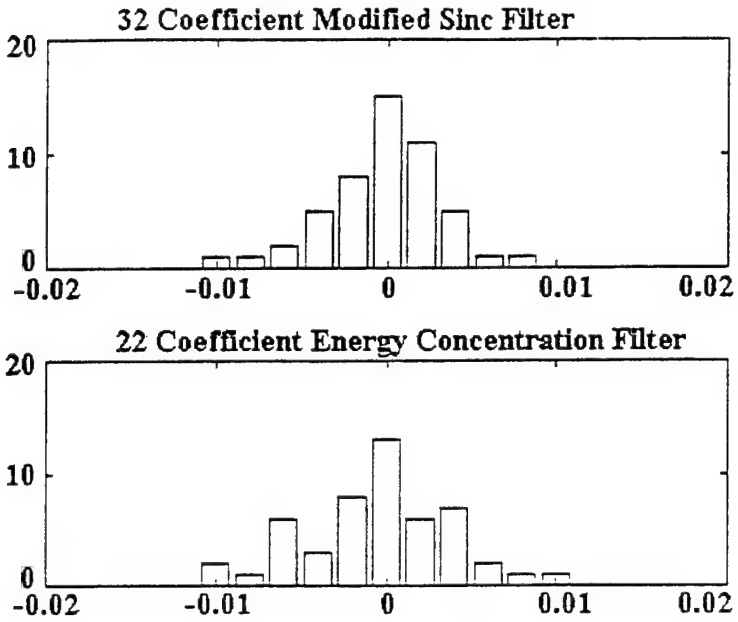



Figure 8.3. Error in Estimates of Hopped/DS Cells
(Horizontal Axis is Hertz, Vertical Axis is the Number of Estimates in Bin)

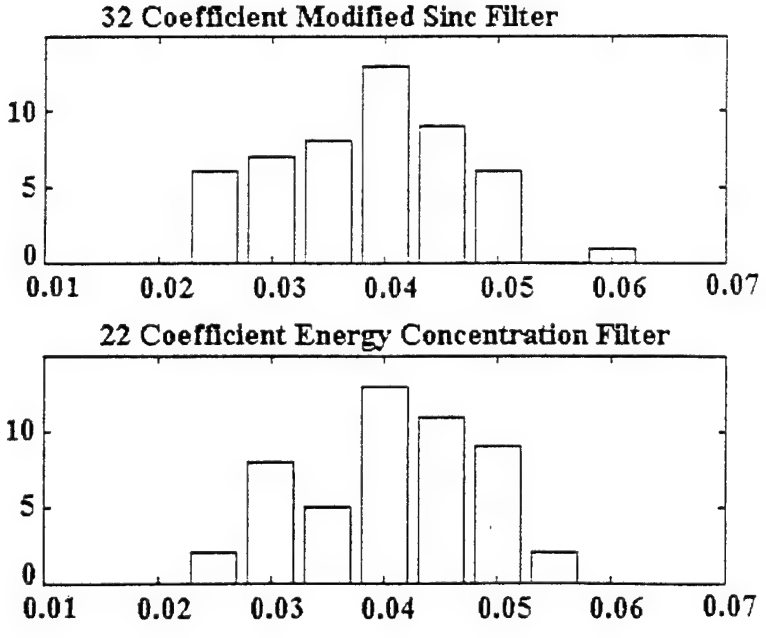



Figure 8.4. Estimates of Bandwidth of Hopped/DS Cells
(Horizontal Axis is Hertz, Vertical Axis is the Number of Estimates in Bin)

This algorithm was discussed in terms of the block's spectral vector. There is no reason, however, why it could not also be used on the temporal vector, in the case of a hop cell whose amplitude is varied under a non-rectangular envelope, when that may be more of a distinguishing characteristic than the distribution of energy in frequency.

One intriguing idea for further research would be to add adaptive techniques to the algorithm. This is similar in concept to digital adaptive equalizers which use least square algorithms to adjust their filter coefficients. In our case, the actual energy observed in the bins of spectral (or temporal) vectors, from blocks containing detected cells, could be used to adjust our expectations for the next observation. The goal would be to "train" the intercept receiver to recognize a transmitter's cells.

One way to accomplish this would be to use an artificial neural net (ANN) as a classifier in the receiver architecture shown in Figure 1.2. In this case, rather than sending the test statistic, z, to the classifier, the entire spectral (and/or temporal) vector would be sent. The ANN could take this as an input, and be trained to recognize vector patterns, although some provision to pre-shift the vector before input, to adjust for variations in the cell's center frequency relative to the block, might be desirable.

# IX. Summary, Conclusions, and Recommendations

## Summary and Discussion

The goal of this research was to develop methods to decompose a waveform, using orthogonal basis functions and a quadrature mirror filter (QMF) bank tree, and extract detailed information about embedded LPI (spread spectrum) signals. In the first chapter, we explored the background of the problem: Defining the specific LPI signals to be used in the analyses, discussing the current state of knowledge in the field of LPI signal detection, and laying out an architecture (Figure 1.2) for an intercept receiver that would accomplish the goal.

The research was primarily concerned with two parts of the receiver architecture: The QMF bank tree, and the analyzer block. Chapters II and III dealt with the former, while Chapters V-VIII discussed algorithms used in the latter.

### QMF Bank Tree

Chapter II was, for the most part, a review and interpretation of published results in the context of our problem. The chapter laid out the general mathematical framework for the QMF bank tree. The decomposition of a function by orthogonal basis functions was discussed, and it was shown how a good choice of basis set could be used to improve the detection of a deterministic signal embedded in white Gaussian noise (WGN). Various examples of orthogonal basis sets were then discussed, leading to the Wavelet Transform and the QMF bank tree. In the course of this discussion, five rules were laid out that would yield finite impulse response (FIR) filters suitable for the tree.

The most important point of the chapter was that, with good filters, the output of the QMF bank tree would be a decomposition of the input waveform in such a way that the square of each value would represent the energy approximately contained in a rectangular tile of the time frequency plane, with each tile having an area of 0.5. Furthermore, the tiles represented by the output of a particular layer of the tree were twice as long in duration, and half as tall in frequency, as those from the layer above.

Although this representation is more-or-less true for any filters meeting the requirements spelled out in the five rules, some filters are better than others at collecting the energy concentrated in the tiles. (It is, of course, mathematically, impossible to do this perfectly.) Somewhat surprisingly, there is little discussion of this specific problem in the Wavelet literature, and so, in Chapter III, several "good" filters were derived.

The first of these was the "modified sinc filter". Basically, it was determined that a tree of filters would concentrate energy perfectly in the frequency dimension if the filters were based on a prototype FIR filter whose coefficients were equal to samples under a sinc function envelope. This basic premise was modified to obtain practical filters meeting our requirements.

Although the modified sinc filter concentrates energy nearly perfectly in the frequency dimension, it does not do as well in the time dimension, and so a compromise filter, trading off the amount of out of tile energy in the time and frequency dimensions, was found. This was done by defining an objective function and minimizing it with respect to the filter coefficients. Although a drawback of this type of filter is that it is optimized for only one particular output of the QMF bank tree, the ability to concentrate energy by other outputs was shown to be good. A function optimizing

the first layer low pass output was found and minimized, and the result was a 22 coefficient "energy concentration filter". Both this filter and the modified sinc filter were used in the analyses described in subsequent chapters.

### Simulation Programs

In the later chapters, Monte-Carlo simulations were used to verify mathematical results and to provide an indication of results when the mathematics, in the analyses, were intractable. Chapter IV described some of the programs used in the simulations. In particular, the code generating the LPI signals was described, as was the code used to decompose a waveform in the manner of the QMF bank tree. Various simple tests, to verify the code functions as it should, were also described.

### Detection

Chapters V-VII developed the analyzer block algorithms to detect and extract features of LPI signals. When the signals included hopping in their structure, as they did in Chapter V (hopped signal cells with time bandwidth products of one) and Chapter VII (hopped cells spread with a DS signal, and slow FH cells), the best method of detection was shown to be the "block algorithm" in which the energy from adjacent tiles (forming blocks) are summed.

The basic premise for this algorithm was laid out in Chapter V, where it was shown that the radiometer/threshold detector is guaranteed to be the best detection architecture only when no assumptions are made about the distribution of the signal's energy in the time frequency plane. It has been shown in the literature, and was discussed in the chapter, that when the interceptor knows the channelization and hop synchronization of the signal, the filter bank combiner (FBC) architecture often yields better performance. The key to the FBC's success is that it divides the time frequency plane into blocks matched (in size and position) with the signal cells and detects concentrations of energy. In the problems discussed in this research, however, the interceptor is assumed not to know the channelization or hop synchronization of the signal(s) being detected. To overcome this, the nature of the QMF bank tree output is exploited: Overlapping blocks in the time frequency plane are examined, with the idea that one of these blocks will best cover each cell. Unfortunately, analysis of the block algorithms' performance was found to be mathematically intractable, due to the non-independence of the noise in the overlapping blocks. However, it was pointed out the results can be bounded: Performance should be somewhat poorer than the FBC, but will often be better than a radiometer/threshold detector. This was verified by example, with simulation.

In Chapter V, the "nine tile scheme" was used to look for signals whose cells have a time bandwidth product of one. In this algorithm, the block dimensions were three tiles by three tiles. By looking at multiple layers of outputs from the QMF bank tree, some particular block will always contain most of the energy from each signal cell. These blocks generally contain a larger total energy than those containing noise only, and so a detection decision can be made by taking the larger energy blocks, throwing out the blocks that overlap, and comparing the high energy blocks to a threshold. In the chapter hard thresholds were used, although in the architecture of Figure 1.2 the block energies would be submitted as input to a classifier block, which may use a more sophisticated decision scheme.

In Chapter VII, it was found that, in general, blocks that are equal to or larger than the cell in each dimension are preferable to blocks that are smaller. This was used in the detection of

hopped/DS and slow FH cells--cells whose time bandwidth products are greater than one. In this chapter a layer was selected in which the tiles were smaller, in each dimension, than the cells to be detected. The block size was then determined by finding the number of tiles in each dimension that it would take to equal or exceed the maximum possible cell size. Then, as in the nine tile scheme, the larger energy blocks were found, overlapping blocks were discarded, and a detection decision was made by comparing the large energy blocks against a threshold.

The detection of DS signals was explored in Chapter VI. However, it was quickly pointed out if this is the sole function of the intercept receiver, it would be better to replace the QMF bank tree with a discrete Fourier Transform. The chief advantage of the QMF bank tree is to simultaneously decompose the input waveform with high resolution both in time and frequency. Since the DS signal is assumed to be turned on for the duration of the observation, high resolution in time is not required.

The algorithm to detect DS signals involves taking a high layer of the QMF bank tree (or taking a discrete Fourier Transform), and forming a "spectral vector", in which each element, or bin, represents the energy from a particular narrow frequency band. Then, the spectral vector is convolved with various sized rectangular windows and, for each rectangle size, the high energy positions are saved, and positions that overlap are discarded. For each of these saved positions, a test statistic is computed and used to compare the data from the different sizes of rectangles. To do this, the test statistics are compared among the different sized overlapping positions, and the positions with the highest value are saved, while the overlapping ones are discarded. The result is a list of suspected DS signals with the energy collected (the result of the convolution), along with the sizes and positions of the rectangles. The detection decision is then made by comparing the energy against a threshold. (Once again, a hard threshold in the case of the simulations performed in the chapter, although a more sophisticated scheme involving a classifier and soft thresholds is possible.)

As with the block algorithms, the analysis of this algorithm is mathematically intractable due to the non-independence of the noise collected by overlapping rectangles. The performance, however, can be bounded and will be less than that of a radiometer adjusted to the bandwidth of the DS signal, but may be greater than that of a radiometer adjusted to the entire observation bandwidth.

### Feature Extraction

"Feature Extraction", in this research, specifically meant estimating the energy, bandwidth, and center frequency of each DS signal or hopped cell, and the duration and position in time of each cell. In addition, in Chapter VIII, the distribution of energy in frequency was used as an estimator to determine the type of cell present. The feature extraction algorithms described in Chapters V-VII are extensions of the detection algorithms.

In Chapter V, the locations of the cells (with time bandwidth products of one) were estimated directly from the positions of the blocks determined to contain them. The cell dimensions were estimated from the layer that the block came from, and the cell energy was estimated by subtracting the expected noise energy from the total energy found in the blocks.

In Chapter VI, the size and position of the rectangular windows, used in the detection algorithm described above, gave estimates of the DS signal's bandwidth and center frequency. The amount of energy collected under each rectangle minus the expected noise energy, yielded an estimate of the signal energy.

In Chapter VII, the locations of the cells (with unknown time bandwidth products greater than one) were roughly estimated from the positions of the blocks determined to contain them. These estimates were refined by using variations on the algorithm developed for the DS signal. A spectral vector was formed from the area of the time frequency plane under each block, and the algorithm was used to estimate each cell's bandwidth and center frequency. Similarly, a "temporal vector" was formed from each block, and the convolution of rectangles algorithm was used to estimate the duration and position in time of each cell. The energy for each cell could be estimated either from the total block energy, or from the energy contained in the time frequency plane under the estimated cell position (with the appropriate expected amount of noise energy subtracted off in each case).

In all three chapters, the feature extraction algorithms were verified via simulation. Histograms of the estimates were presented to give the reader a sense of the distribution.

The advantage of using rectangular shaped windows in some of algorithms described above is that the intercept receiver is somewhat less sensitive to the particular distribution of signal energy. This is desirable when the interceptor does not know precisely what distribution to expect, but disregards the possibility of using the distribution as a method to distinguish between transmitters' signal cells. This possibility is explored in Chapter VIII, where the distribution of cell energy in frequency is used to distinguish hopped/DS cells with sinc squared distributions from others.

Specifically, the algorithm calls for the interceptor to first use the block detection algorithm as described in Chapter VII. Spectral vectors are then computed from the blocks determined to contain cells. "Expected spectral vectors" are then computed from the expected energy in each bin of the spectral vector for the possible bandwidths and cell positions. These are compared against the measured spectral vector for each block by summing the squared difference for each bin. The minimum of these values is taken to be from the best match of expected and true bandwidth and position. This value is then compared against a threshold and, if less, the decision is made that the cell has the assumed distribution.

This algorithm also offers an alternative method, to the one in Chapter VII, for estimating a cell's center frequency and bandwidth. Simply: The best match of expected spectral vector with the true vector should occur when the center frequency and bandwidths used to compute the expected spectral vector are closest to matching the true values.

### Combining the Algorithms

In the event the interceptor wishes to simultaneously look for different types of spread spectrum signals, there is no reason the algorithms discussed in this paper can not be combined. At the risk of boring the reader by repeating information, here is a concise summary of a combined algorithm for the analyzer block in the receiver architecture of Figure 1.2:

1) Form a spectral vector of the entire observed time frequency plane using the output from the highest layer of the QMF bank tree (yielding the finest resolution). From this, use the algorithm described in Chapter VI to look for DS signals. Namely: Form rectangular windows whose sizes range across the possible signal bandwidths, and convolve them with the spectral vector. Save the largest elements of convolution--representing, presumably, the energy collected when the rectangular window best covers a DS signal--and throw out elements from overlapping rectangles. The resulting list with energies, rectangle sizes, and positions, can then be passed to the classifier to make detection decisions. (When the interceptor knows the shape of the spectrum of signals to be detected, the least

squares algorithm of Chapter VIII may also be used, in addition to, or in lieu of, the one from Chapter VI.)

2) Pick an intermediate layer of the QMF bank tree output to look for hopped signals whose cells have time bandwidth products greater than one. The layer should have tiles that are smaller than the maximum cell dimensions in both time and frequency. Then perform the block detection algorithm described in Chapter VI: From the intermediate layer, form overlapping blocks, with the block dimensions equal, in each case, to the maximum possible cell dimension. Look for blocks containing the highest amounts of energy, and throw out overlapping blocks. The resulting list of blocks, with their locations and the amounts of energy, can then be passed to the classifier to make the detection decision.

When a large range of cell sizes are looked for, detection performance can be improved by repeating 2) with smaller block sizes.

Once the blocks are found using 2), the cell dimensions and positions can be estimated by forming spectral and temporal vectors from the blocks (using, respectively, high and low layers of the QMF bank tree to improve resolution) and convolving them with rectangular windows, in a manner similar to that described in 1) above. Results can be passed to the classifier along with the list of blocks.

Alternatively, the least squares algorithm discussed in Chapter VIII may be used to find the cells' center frequency and bandwidth, when the interceptor knows the cells' spectral shape. This algorithm may also be used to distinguish between cells.

3) For hopped cells with time bandwidth products of one, carry out the detection algorithm discussed in Chapter V: Select a range of QMF bank tree output layers such that the frequency dimension of three by three tile blocks exceeds the maximum cell bandwidth for the lowest layer, and the time dimension exceeds the duration of the cell at the highest layer. For each of these layers, find the three by three blocks containing the maximum energy and save those while discarding overlapping blocks. Then repeat this, comparing the blocks from the different layers. The resulting list of block energies, sizes, and positions, can then be passed to the classifier to make a detection decision and to estimate the cell features.

## Major Conclusions and Contributions of the Research

This research establishes that:

1) The overall architecture of the intercept receiver presented in this paper, as shown in Figure 1.2, is sound. It is a good alternative to the simple radiometer/threshold detector, particularly for detecting hopped LPI spread spectrum signals when the channelization, cell duration, and hop synchronization are unknown.

2) Both the modified sinc filter and energy concentration filter yield good results when used in the QMF bank tree.

3) The nine tile scheme appears to yield good detection performance against cells with time bandwidth products of one. Although it is dangerous to generalize too widely from simulation results, they do seem to indicate the nine tile scheme will out-perform the radiometer/threshold detector in many situations.

4) The QMF bank tree appears not to be the best method for decomposing an input signal when the only goal is the detection of DS signals.

5) For detecting hopped signals whose cells' time bandwidth products are greater than one, the block detection algorithm, as laid out in Chapter VII, yields good performance results. Simulations indicate this exceeds the performance for the radiometer/threshold detector.

6) By examining the dimensions of the blocks resulting from the nine tile scheme, the dimensions of cells with time bandwidth products equal to one may be estimated. These estimates should be close enough, in many cases, for a classifier to determine which cells came from separate transmitters. Similarly, each nine tile block's energy provides a good estimate of the signal cell's energy.

7) Estimates of DS signals' center frequencies and bandwidths may be made with the algorithm described in Chapter VI. A similar algorithm may be used to estimate cell sizes and locations within blocks, for blocks found using the algorithm described in Chapter VII for detecting cells with time bandwidth products greater than one. The estimates of position and time duration are generally good, although the estimates of bandwidth may leave something to be desired. In the case where the interceptor knows the general spectrum of the signal energy, the least squares algorithm described in Chapter VIII appears to offer better results.

8) Chapter VIII's least squares algorithm is a good method to distinguish transmitters' cells when the cells' spectral shapes differ significantly.

9) When desirable, the various algorithms discussed in this paper may be combined as discussed in the last section above.

## Recommendations For Further Research

The research reported in this paper has explored the possibilities of using QMF bank trees to decompose a received waveform to look for and detect LPI spread spectrum signals. As such, it has necessarily been rather broad in scope, and, because of this, most, if not all, of the areas discussed in this paper could be examined in greater depth. What follows is a brief list of ideas that have occurred to the author during the course of investigation. The reader may, of course, have further inspiration or insight.

1) Although the performance evaluation of algorithms involving overlapping blocks appears to be mathematically intractable (see Chapter V), additional work in this area may prove valuable. Tighter bounds, or suitable approximations, yielding detection probabilities for given signal parameters, would be a great aid to the engineer in determining if this intercept receiver is appropriate for a particular scenario. This appears to be a problem in non-independent order statistics.

2) In this research, whenever signal energy was treated as a random variable, only the expected values were used in the analysis. For example, in Chapter VIII, the expected signal energy in each bin of the spectral vector was used. Further insight may be gained by determining (or approximating) the pdf, or at least the second moments. Even more valuable would be to extend this analysis and, look not just at the pdf of signal energy in each spectral bin, but in each tile of the time frequency plane. In the research in this paper, the energy from each signal cell contained in a block

was taken to be more-or-less deterministic. An analysis determining the pdf of the energy in each tile might well lead to better detection algorithms, and additional methods for distinguishing between multiple signals.

3) Including the classifier in the analysis is an obvious next step. Especially in some particular scenario, it would be interesting to see how well an adaptive classifier, such as an artificial neural network, can use the information output from the analyzer algorithms to detect and classify signals. Including the relatively new developments in "fuzzy logic" adds even more possibilities.

4) Cyclostationary detectors have not been discussed in this paper with the exception of a very brief mention in the literature review in Chapter I. In this sort of detector, the spectral correlation of the input waveform is computed, and examined for periodicities that may be due to an LPI signal [22] [23]. It would be interesting to compare the capabilities and performance of this method to the ones discussed in this paper. It may also be possible to combine some of the concepts presented here, particularly the QMF bank tree, with cyclostationary detection.

5) Chapter VIII explored the possibilities of distinguishing between different transmitter's cells based on the cells' spectral shapes. There, hopped/DS cells were compared to the radically different slow FH cells. It would be interesting to see well this method works for distinguishing between more subtle differences in energy distribution. For example: In the hopped/DS cells considered in this paper, the transmitter was assumed to turn the cell on and then off instantaneously. This, of course, is not possible in reality--any real transmitter will broadcast transients shaping the cell in some manner. One possibility for further research would be to determine whether individual transmitters might be distinguished from one another based on these transients. In addition to looking at the spectral vectors, as in Chapter VII, temporal vectors may well prove worthy of analysis in this effort. An adaptive classifier would also probably be vital here, since it could attempt to find and match subtle differences in cell shapes without a priori knowledge.

## Appendix A: Matlab Files Described in Chapter IV

### Table of Contents

```
function[n] = noisegen(seed)
%
%  [n] = noisegen(seed)          Generates a file of 32768 elements
%                                of Gaussian distributed noise with
%                                mean = 0, var = 1
%
%  seed     is the seed value

NUM_SAMPLES = 32768;

randn('seed',seed);

n = randn(NUM_SAMPLES,1);


function [signal] = ds(seed,theta,fc,B)
%
%     [signal] = ds(seed,theta,fc,B)
%
%     The signal is assumed to be sampled at one sample/sec, so
%     the normalized digital frequency range is [0,1/2).
%
%     INPUTS:
%     seed    seed for DS spreading
%     theta   carrier offset
%     fc      center frequency of signal
%     B       bandwidth of signal
%
%     Both fc and B are digital frequencies
%
%     OUTPUT:
%     signal  the signal sequence

NUM_SAMPLES = 32768;
t = [1:NUM_SAMPLES];

% A is a random binary waveform that varies from +1 to -1
rand('seed', seed);
bin = floor(2 .*rand(ceil(NUM_SAMPLES.*B),1));
A = 2 .*bin(ceil(t(:).*B)) - 1;

signal = A(:).*sin(2 .*pi.*t(:).*fc + theta);
```

```
% dsdemo.m  Script file to demonstrate ds.m

[signal] = ds(86,0,.25, .15);

M = stft(signal,128,128);
M = M.*conj(M);
[m,n] = size(M);
M = M([1:m./2],:);
contour(M);

figure;
f = fft(signal);
f = f.*conj(f);
f = f([length(f)./2:length(f)]);
plot([1:length(f)],f);


function [signal, channel] = ffh(seed1,seed2,theta,lf,CH,T,phase,B)
%
%     [signal, channel] = ffh(seed1,seed2,theta,lf,CH,T,phase,B)
%                         Fast Frequency Hopped Signal.
%
%     The signal is assumed to be sampled at 1 sample/sec, so the
%     normalized digital frequency range is [0,1/2).
%
%     INPUTS:
%     seed1     seed for random hop sequence
%     seed2     seed for DS spreading
%     theta     carrier phase
%     lf        lowest frequency (lf = center freq of channel 1 - B/2)
%     CH        number of channels
%     T         cell length (in samples)
%     phase     shifts the cells by <phase> samples, range:  [0,T)
%     B         (optional) cell bandwidth
%
%     If B is not present, the signal generated will be pure FH.
%     Otherwise, each cell is spread with a DS sequence.
%     lf and B are both digital frequencies.
%
%     OUTPUTS:
%     signal    the signal sequence
%     channel   the hop sequence

NUM_SAMPLES = 32768;
t =[1:NUM_SAMPLES];

% Select channels randomly -- channel will be the hop sequence,
% its length will equal the number of cells
rand('seed', seed1);
channel = ceil(CH.*rand(ceil(NUM_SAMPLES./T)+1,1));
```

```
% Spread cells with a random binary waveform
% A is the waveform and varies between +1 and -1
if nargin < 8
    B = 1 ./T;
    A = ones(NUM_SAMPLES,1);
else
    rand('seed', seed2);
    bin = floor(2 .*rand(ceil(NUM_SAMPLES.*B),1));
    A = 2 .*bin(ceil(t(:).*B)) - 1;
end;

% cell_number is a sequence equal in length to t
% It indicates which cell is on at which time
cell_number(:) = (floor((t(:)+phase-1)./T) + 1)';

signal = A(:).*sin(2 .*pi.*t(:).* ..
         (1f B./2+B.*channel(cell_number(:)))+theta);


% ffhdemo.m  Script file to demonstrate ffh.m

[signal, channel] = ffh(10,57,0,0,20,3200,0,.025);

M = stft(signal,128,128);
M = M.*conj(M);
[m,n] = size(M);
M = M([1:m./2],:);
contour(M);


function [signal, position] =
th(seed1,seed2,theta,fc,slots,T,phase,B)
%
%     [signal, position] = th(seed1,seed2,theta,fc,slots,T,phase,B)
%                          Time Hopped Signal
%
%     The signal is assumed to be sampled at 1 sample/sec, so the
%     normalized digital frequency range is [0,1/2).
%
%     INPUTS:
%     seed1     seed for the random position sequence
%     seed2     seed for DS spreading
%     theta     carrier phase
%     fc        center frequency
%     slots     number of slot positions per transmitted cell
%     T         cell length (in samples)
%     phase     shifts the cells by <phase> samples,
%               range:  [0,T*slots)
%     B         (optional) cell bandwidth
%
%     If B is not present, the signal generated will be pure TH.
%     Otherwise, each cell is spread with a DS sequence.
%     fc and B are both digital frequencies.
%
%     OUTPUTS:
%     signal    the signal sequence
%     position  the position sequence
```

```
NUM_SAMPLES = 32768;
t = [1:NUM_SAMPLES];

% Select cell positions randomly
rand('seed',seed1);
spc = slots.*T;                    % samples per cell
N = ceil(NUM_SAMPLES./spc + 1);    % Number of cells
position = ceil(slots.*rand(N,1));

% TX will be used to turn the signal on during transmit positions
TX = zeros(NUM_SAMPLES + spc,1);
on = ones(T,1);
for i = 1:N
      % j is the position of the 1st sample of a cell
      j = (i-1).*spc + T.*(position(i)-1) + 1;
      TX(j:(j+T-1),1) = on(:,1);
end;
TX = TX((1 + phase):(NUM_SAMPLES + phase),1);

% Spread cells with a random binary waveform
% A is the waveform and varies between +1 and -1
if nargin < 8
    B = 1 ./T;
    A = ones(NUM_SAMPLES,1);
else
    rand('seed', seed2);
    bin = floor(2 .*rand(ceil(NUM_SAMPLES.*B),1));
    A = 2 .*bin(ceil(t(:).*B)) - 1;
end;

signal = A(:).*TX(:).*sin(2 .*pi.*t(:).*fc + theta);


%  thdemo.m  script file to demonstrate th.m

[signal, position] = th(10,57,0,.25,3,3200,0,.025);

M = stft(signal,128,128);
M = M.*conj(M);
[m,n] = size(M);
M = M([1:m./2],:);
contour(M);
```

```
function [signal,channel,position] = ..
         ffhth(seed1,seed2,seed3,theta,lf,CH,slots,T,phase,B)
%
%      [signal,channel,position] =
%           ffhth(seed1,seed2,seed3,theta,lf,CH,slots,T,phase,B)
%           Fast Frequency Hopped/Time Hopped Signal.
%
%      The signal is assumed to be sampled at 1 sample/sec, so the
%      normalized digital frequency range is [0,1/2).
%
%      INPUTS:
%      seed1     seed for the random position sequence
%      seed2     seed for random hop sequence
%      seed3     seed for DS spreading
%      theta     carrier phase
%      lf        lowest frequency (lf = center freq of channel 1 - B/2)
%      CH        number of channels
%      slots     number of slot positions per transmitted cell
%      T         cell length (in samples)
%      phase     shifts the cells by <phase> samples, range:
%                [0,T*slots)
%      B         (optional) cell bandwidth
%
%      If B is not present, the signal generated will be pure FH.
%      Otherwise, each cell is spread with a DS sequence.
%      lf and B are both digital frequencies.
%
%      OUTPUTS:
%      signal    the signal sequence
%      channel   the hop sequence
%      position the position sequence

NUM_SAMPLES = 32768;
t = [1:NUM_SAMPLES];

% Select cell positions randomly
rand('seed',seed1);
spc = slots.*T;                      % samples per cell
N = ceil(NUM_SAMPLES./spc + 1);      % Number of cells
position = ceil(slots.*rand(N,1));

% TX will be used to turn the signal on during transmit positions
TX = zeros(NUM_SAMPLES + spc,1);
on = ones(T,1);
for i = 1:N
      % j is the position of the 1st sample of a cell
      j = (i-1).*spc + T.*(position(i)-1) + 1;
      TX(j:(j+T-1),1) = on(:,1);
end;
TX = TX((1 + phase):(NUM_SAMPLES + phase),1);
```

```
% Select channels randomly -- channel will be the hop sequence,
% its length will equal the number of cells
rand('seed', seed2);
channel = ceil(CH.*rand(ceil(NUM_SAMPLES./spc)+1,1));

% Spread cells with a random binary waveform
% A is the waveform and varies between +1 and -1
if nargin < 10
    B = 1 ./T;
    A = ones(NUM_SAMPLES,1);
else
    rand('seed', seed3);
    bin = floor(2 .*rand(ceil(NUM_SAMPLES.*B),1));
    A = 2 .*bin(ceil(t(:).*B)) - 1;
end;

% cell_number is a sequence equal in length to t
% It indicates which cell is on at which time
cell_number(:) = (floor((t(:)+phase-1)./spc) + 1)';

signal = A(:).*TX(:).* ..
       sin(2 .*pi.*t(:).*(lf-B./2+B.*channel(cell_number(:))) +...
           theta);


%  fhthdemo.m  script file to demonstrate ffhth.m

[signal,channel,position] = ffhth(7,47,43,0,0,20,3,3200,0,.025);

M = stft(signal,128,128);
M = M.*conj(M);
[m,n] = size(M);
M = M([1:m./2],:);
contour(M);
```

```
function [signal, channel] =
sfh(seed1,seed2,theta,lf,CH,T,phase,M,Ntpc)
%
%     [signal, channel] = sfh(seed1,seed2,theta,lf,CH,T,phase,M,Ntpc)
%                         Slow Frequency Hopped Signal
%
%     The signal is assumed to be sampled at 1 sample/sec, so the
%     normalized digital frequency range is [0,1/2).
%
%     INPUTS:
%     seed1     seed for random hop sequence
%     seed2     seed for information sequence
%     theta     carrier phase
%     lf        lowest frequency
%     CH        number of channels
%     T         cell length (in samples)
%     phase     shifts the cells by <phase> samples, range:  [0,T)
%     M         number of FSK frequencies
%     Ntpc      number of FSK transitions per cell
%
%     OUTPUTS:
%     signal    the signal sequence
%     channel   the hop sequence

NUM_SAMPLES = 32768;
t = [1:NUM_SAMPLES];
spt = T./Ntpc;  % samples per transition
bw = 1 ./spt;   % bandwidth of symbol
B = M.*bw;      % cell bandwidth

% randomly generate an information sequence
rand('seed', seed2);
info = ceil(M.*rand(ceil(NUM_SAMPLES./spt + Ntpc),1));

% Select channels randomly -- channel will be the hop sequence,
% its length will equal the number of cells
rand('seed', seed1);
channel = ceil(CH.*rand(ceil(NUM_SAMPLES./T)+1,1));

% cell_number is a sequence equal in length to t
% It indicates which cell is on at which time
cell_number(:) = (floor((t(:)+phase-1)./T) + 1)';

% symbol is a sequence equal in length to t
% It indicates which information symbol is on at which time
symbol(:) = ceil((t(:)+phase-1+eps)./spt)';

% frequency at each sample time
freq(:) = lf - B + ..
          B.*channel(cell_number(:)) - bw./2 + bw.*info(symbol(:));

signal = sin(2 .*pi.*t(:).*freq(:) + theta);
```

```
% sfhdemo.m  Script file to demonstrate sfh.m

[signal, channel] = sfh(10,57,0,0,20,3200,0,4,10);

M = stft(signal,128,128);
M = M.*conj(M);
[m,n] = size(M);
M = M([1:m./2],:);
contour(M);


function qmf(f,filter,N)
%    qmf(f,filter,N)
%
%     Quadrature Mirror Filter.  This takes the input waveform f, and
%     processes it with a QMF bank.  'filter' is optional and
%     specifies a file of filter coefficents.  Default is haar.m
%
%     Modification:  19 Feb 95. N is an optional input and specifies
%     the last layer to be computed.  This can save time if the
%     higher layers are not needed.  The default is to compute all of
%     the layers (this provides backward compatibility).

if nargin < 2
      filter = 'haar.m';
end;

n = floor(log2(length(f)));

% number of layers
if nargin < 3
      N = n;
end;

I([1:2 .^n],1) = f([1:2 .^n]);
out = I;
```

```
% decompose the function
for lay = 1:N     % layer

  disp(lay)
  flag = 1;

  % reshape the output matrix
  [r,c] = size(out);
  out = zeros(r./2, c.*2);

  for i = 1:2 .^(lay-1)              % column of I (low to high)
      [G,H] = feval(filter, I(:,i));

      if flag
            out(:, i.*2-1) = H;
            out(:, i.*2) = G;
      else
            out(:, i.*2-1) = G;
            out(:, i.*2) = H;
      end;
            flag = ~flag;
  end;

  I = out;

  % save the data to disk
  eval(['save c:\data\layer',int2str(lay),'.dat I /ascii /double']);

end;
```

```
function [d1, c1] = haar(c0)
%
% [d1, c1] = haar(c0)  uses Haar filter to decompose the column
%                      vector c0 into a (high frequency) column
%                      vector d1 and(low frequency) column vector c1.

% Daubechies filter coefficients
h0 = 2 .^(-1/2);
h1 = h0;

h = [h0 h1];
g = [h1 -h0];

N = length(c0);

% i will decimate by 2
i = 2:2:N;

% compute c1
c1 = filter(fliplr(h),1,c0);

% decimate
c1 = c1(i);

% compute d1
d1 = filter(fliplr(g),1,c0);

% decimate
d1 = d1(i);



function [d1, c1] = da4(c0)
%
% [d1, c1] = da4(c0)  uses Daubechies' 4 coefficient Wavelet filter
to
%                     decompose the column vector c0 into a (high
%                     frequency) column vector d1 and (low frequency)
%                     column vector c1.

% Daubechies filter coefficients
h0 = .482962913145;     % main coefficient
h1 = .836516303738;     % main coefficient
h2 = .224143868042;
h3 = -.129409522551;

h = [0 0 h0 h1 h2 h3];
g = [h3 -h2 h1 -h0 0 0];

N = length(c0);

% pad with zeros to clear out filter
pad = 4;
c0 = [c0; zeros(pad,1)];

% i will decimate by 2
i = pad:2:(N+pad-2);

% compute c1
c1 = filter(fliplr(h),1,c0);
```

```
% decimate
c1 = c1(i);

% compute d1
d1 = filter(fliplr(g),1,c0);

% decimate
d1 = d1(i);



function [d1, c1] = da16(c0)
%
% [d1, c1] = da16(c0) to uses Daubechies' 16 coefficient Wavelet
%                     filter to decompose the column vector c0 into a
%                     (high frequency) column vector d1 and (low
%                     frequency)column vector c1.

% Daubechies filter coefficients
h0 = .054415842243;
h1 = .312871590914;
h2 = .675630736297;     % main coefficient
h3 = .585354683654;     % main coefficient
h4 = -.015829105256;
h5 = -.284015542962;
h6 = .000472484574;
h7 = .128747426620;
h8 = -.017369301002;
h9 = -.044088253931;
h10 = .013981027917;
h11 = .008746094047;
h12 = -.004870352993;
h13 = -.000391740373;
h14 = .000675449406;
h15 = -.000117476784;

h = [zeros(1,10) h0 h1 h2 h3 h4 h5 h6 h7 h8 h9 ...
                                h10 h11 h12 h13 h14 h15];
g = [h15 -h14 h13 -h12 h11 -h10 h9 -h8 h7 -h6 ...
                h5 -h4 h3 -h2 h1 -h0 zeros(1,10)];

N = length(c0);

% pad with zeros to clear out filter
pad = 14;
c0 = [c0; zeros(pad,1)];


% i will decimate by 2
i = pad:2:(N+pad-2);

% compute c1
c1 = filter(fliplr(h),1,c0);

% decimate
c1 = c1(i);
% compute d1
d1 = filter(fliplr(g),1,c0);
% decimate
d1 = d1(i);
```

```
function [d1, c1] = con22(c0)
%
% [d1, c1] = con22(c0)  uses the energy concentrating 22 coefficient
%                       Wavelet filter to decompose the column vector
%                       c0 into a (high frequency) column vector d1
%                       and (low frequency) column vector c1.

% filter coefficients

h0 =  -0.06937058780687;
h1 =   0.08787454185760;
h2 =   0.69303661557137;      % main tap
h3 =   0.69307535109821;      % main tap
h4 =   0.09491659263189;
h5 =  -0.09139339673362;
h6 =  -0.04245335524319;
h7 =   0.04675991002879;
h8 =   0.03613538459682;
h9 =  -0.03468231058910;
h10 = -0.02281230449607;
h11 =  0.02100935003910;
h12 =  0.02296583051731;
h13 = -0.02145780458290;
h14 = -0.01348759528448;
h15 =  0.01318436272982;
h16 =  0.01550256127783;
h17 = -0.01636308107201;
h18 = -0.00627176286924;
h19 =  0.00993238693842;
h20 = -0.00105459770882;
h21 = -0.00083252852776;
h = [zeros(1,16) h0 h1 h2 h3 h4 h5 h6 h7 h8 h9 ...
     h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21];
g = [h21 -h20 h19 -h18 h17 -h16 h15 -h14 h13 -h12 h11 ...
     -h10 h9 -h8 h7 -h6 h5 -h4 h3 -h2 h1 -h0 zeros(1,16)];

N = length(c0);

% pad with zeros to clear out filter
pad = 20;
c0 = [c0; zeros(pad,1)];

% i will decimate by 2
i = pad:2:(N+pad-2);

% compute c1
c1 = filter(fliplr(h),1,c0);

% decimate
c1 = c1(i);

% compute d1
d1 = filter(fliplr(g),1,c0);

% decimate
d1 = d1(i);
```

```
function tsinc_su(N);
%
%  tsinc_su(N)    finds the coefficients for the truncated sinc
%                 filters.  These are saved in the data directory
%                 under h.dat (lpf) and g.dat (hpf).  The Hamming
%                 window is used to suppress the Gibb's phenomena.
%
%                 N is the number of coefficients.  If N is between
%                 16 and 32768, the sinc is compressed slightly to
%                 eliminate the loss of energy in the filter tran-
%                 sitions.  For best results, N should be a power
%                 of two.

tab1 = [16   1.80745449451012
        32   1.90189442447347
        64   1.95048099151118
        128  1.97512319592231
        256  1.98753217974055
        512  1.99375872328059
        1024 1.99687751845516
        2048 1.99843829823671
        4096 1.99921903384616
        8192 1.99960948810196
        16384 1.99980473684531
        32768 1.999902366662119];

tab2 = [16   1.22099338489190
        32   1.10429475163860
        64   1.05069751126202
        128  1.02499785161815
        256  1.01241259545240
        512  1.00618488680080
        1024 1.00308711196069
        2048 1.00154222577172
        4096 1.00077078066449
        8192 1.00038530731821
        16384 1.00019263291076
        32768 1.00009631126893];

if (N < 16) | (N > 32768)
        C = 2;
        S = 1;
else
        C = table1(tab1,N);
        S = table1(tab2,N);
end;

x = [-floor(N./2):floor(N./2)-1]';
h = sqrt(S./2).*sinc((x + .5)./C);

% window
w = hamming(N);
h = w.*h;
% find hpf
g = h;
g([2:2:N]) = -h([2:2:N]);

% save data
save c:\data\h.dat h /ascii /double;
save c:\data\g.dat g /ascii /double;
```

```
function [d1, c1] = tsinc(c0)
%
% [d1, c1] = tsinc(c0)  uses truncated sinc filter coefficients to
%                       decompose the column vector c0 into a (high
%                       frequency) column vector d1 and (low
%                       frequency) column vector c1.
%
%     tsinc_su must have been run previously to create h.dat and
%     g.dat.  That program determines how many coefficients are used.

% load filter coefficients
load h.dat;  h = h';
load g.dat;  g = g';

N = length(c0);

% pad with zeros to clear out filter
pad = length(h)./2 + 1;
c0 = [c0; zeros(pad,1)];

% i will decimate by 2
i = pad:2:(N+pad-2);

% compute c1
c1 = filter(fliplr(h),1,c0);

% decimate
c1 = c1(i);

% compute d1
d1 = filter(fliplr(g),1,c0);

% decimate
d1 = d1(i);


function[signal] = tone(amp, df, theta);
%
%     [signal] = tone(amp, df, theta)  generates a tone.
%
%     amp   amplitude
%     df    digital frequency  (0 to 1/2)
%     theta phase (in radians)

NUM_SAMPLES = 2 .^15;
t = 1:NUM_SAMPLES;

signal = amp.*sin(2 .*pi.*t.*df + theta)';
```

```
% ttgendat.m      script file to generate qmf test data for a tone

df = 0.3;
amp = 1;
theta = 0;

tt = tone(amp, df, theta);
qmf(tt,'da16');
clear tt;


% discon.m     script file to display a contour plot of the
%              output of a layer of the qmf bank

load layer7.dat;
M = layer7 .^2;
clear layer7;

contour(M')


% crosst.m   shows a section of the qmf output data taken halfway
%            through the observation period.

load layer10.dat
[m,n] = size(layer10);
dat = layer10(ceil(m./2),:).^2;

clear layer10;

plot(1:n,dat);
axis([1 n 0 (1.1).*max(dat)]);


% imptest.m  script file to generate qmf test data for an impulse

length = 2 .^15;
f = zeros(length,1);
f(length./2) = 1;
qmf(f, 'haar');
clear f;


% crossf.m   shows sections of the qmf output data taken at
%            all frequencies

load layer10.dat
[m,n] = size(layer10);
dat = layer10 .^2;

clear layer10;

plot(1:m,dat(1:m,:), 'r');
axis([1 m 0 (1.1).*max(max(dat))]);
```

```
% noistest.m      script file to test linearity and
%                 Parceval relationship, with noise

seed = 8016;
noise = noisegen(seed);
qmf(noise, 'haar',1);

en0 = sum(sum(noise.^2));
load layer1.dat;
en1 = sum(sum(layer1 .^2));

disp('(energy in-energy out)/energy in')
disp((en0 - en1)/en0)

ci = normgof(noise);
disp('noise:')
disp(ci)
disp(mean(noise))
disp(var(noise))

ci = normgof(layer1(:));
disp('layer1:')
disp(ci)
disp(mean(layer1(:)))
disp(var(layer1(:)))


function [ci] = normgof(V)
%
%  [ci] = normgof(V)  Performs a Chi-Squared Goodness of Fit Test on
%                     the elements of vector V to see if they are
%                     from a Gaussian distributed random process with
%                     mean = 0 and variance = 1.
%
%                     V is the input vector.  The elements are placed
%                     into 8 bins and results are compared with a
%                     Chi-squared variable with 7 degrees of freedom.
%
%                     The output, ci, is a 0-1 vector giving the
%                     results of the following confidence tests:
%
%                     .995  .990  .975  .950  .050  .025  .010  .005
%
%                     Gaussian and Chi-Squared results come from the
%                     Appendicies in Random Signals, Detection,
%                     Estimation, and Data Analysis by Shanmugan and
%                     Breipohl

% Percentage Points on Chi-Squared Distribution with 7 degrees of
% freedom
cs = [.989265 1.239043 1.68987 2.16735 14.0671 16.0128...
      18.4753 20.2777];

% Set up bins
bin = [-1.43 -.87 -.48 -.16 .16 .48 .87 1.43];

% bin data
x = hist(V,bin)';
```

```
% expected
ex = (0.125).*length(V);

% test variable
v = sum(((ex - x(:)).^2)./ex);

% compare test variable to each confidence value
ci = (v.*ones(1,length(cs))) <= cs;
```

## Appendix B: Matlab Files Used to Carry Out Simulations Presented in Chapter V

### Table of Contents

QMFFBC.M decomposes a waveform with the QMF bank tree and evaluates the output as a filter bank combiner (FBC) receiver. The first input is the waveform, assumed to consist either of WGN, or of WGN and a Fast FH signal with cells 128 samples long each in 32 channels from 0.125 to 0.375 Hz. The other two inputs are the thresholds, **Th** and **K**, shown in Figure 5.6. **Th** may be a vector of threshold values yielding a vector of corresponding results. The advantage of this will be seen when the simulation is described.

The file first evaluates the waveform with QMF.M, then finds the energy in the tiles at layer six from 0.125 to 0.375 Hz. Energy in pairs of tiles adjacent in time are then added to create a matrix **ro**. The values of **ro** are then compared to each threshold in **Th**, and the results are evaluated in a manner logically identical to the OR gate and BMW. Finally, each element in the binary output vector, **det**, is assigned "1" if a signal is taken to be present and "0" otherwise, for each corresponding threshold in **Th**.

```
function det = qmffbc(waveform,Th,K)
%       det = qmffbc(waveform,Th,K)    Uses the QMF as a FBC to
%                                      evaluate a waveform
%
%       The signal is assumed to be FH with cell length = 128 and with
%       32 channels
%
%       Th is the threshold on the radiometers, and may be a vector.
%       K is the threshold on the BMW and must be a number.
%       det is a binary output vector and is 1 if a signal is assumed
%           to be present.

qmf(waveform,'con22',6);

load layer6.dat
[m,n] = size(layer6);
dat = layer6 .^2;
clear layer6;
% only look at frequencies w channels
dat = dat(1:m,(n./4+1):(3.*n./4));
% combine signal as in a FBC
[m,n] = size(dat);
ro = dat(1:2:(m-1),:) + dat(2:2:m,:);

% threshold
for i = 1:length(Th)
      L = ro > Th(i);
      x = sum(any(L'));
      det(i) = x > K;
end;
```

Simulations of an FBC were carried out with the script file FBCSIM.M. As described in Chapter V, each "set" consists of 100 runs with noise only, and 100 runs of signal and noise. To do this, FBCSIM.M was called separately for each 100 runs. For noise only inputs, the line determining the signal was commented out, and the line constructing **waveform** was modified. The signal in this case is created by FFHTH.M (described in Chapter IV and listed in Appendix A). Each cell is located among 32 channels and 25 time slots. The channels range from 0.125 to 0.375 Hz. Each time slot is 128 seconds long. In the 32768 second observation period, we should see 10 cells, and will occasionally see the first part of another. To obtain a total cell energy of 20, the signal is multiplied by 0.559, as indicated by (5.1).

```
% fbcsim   script file to simulate using the qmf bank as a FBC

start = clock;

Th = [16:0.2:23];
K = 1;
rep = 100;

for loop = 1:rep
      disp(loop)
      % generate signal
      [signal,channel] = ...
         ffhth(800000+loop,500000+loop,67,0,0.125,32,25,128,0);
      noise = noisegen(loop+100000);
      waveform = noise + (.559).*signal;

      det(loop,:) = qmffbc(waveform,Th,K);
end;

P = sum(det)./rep;

disp(etime(clock,start));
```

TILE.M carries out the "nine tile scheme" as described in Chapter V. The input, **M**, is a matrix of the squared coefficients from a layer of the QMF bank. Following the code listing: **bf** and **bt** are the block dimensions in frequency and time (changing these are all that is necessary to change this from a "nine" tile scheme to one with other block dimensions), and **blt** and **blf** are the number of blocks along each dimension in the time frequency plane. The matrix **block**, with dimensions **blt** and **blf**, is allocated, and will be used to hold the energy for each 3 by 3 block.

The energy for each block is then found by iterating through a nested loop, with **i** being the time index and **j** the frequency. **X** is a 3 by 3 matrix taken from the appropriate portion of **M**, and containing the energy coefficients for the 9 tiles in each block. These values are summed, and the result is stored as the appropriate element in **block**.

**cell_list** will be the eventual output for the file, and is a three column list. The first column will contain block energy, while the second and third columns will contain the blocks' positions in time and frequency, respectively. Since the final length of **cell_list** will depend on how the block energy is distributed and the way blocks overlap, we cannot know how long the list will be, but we know it will be at least nine times (**bt** times **bf**) smaller than the number of elements in **block**, so this amount of space is (over) allocated for **cell_list**.

Rather than sort the values in **block**, then go through the list looking for overlaps, as described in the strategy sub-section above, it was found to be faster (and is logically equivalent) to retain **block** as is, use the built in "max" command to find the largest element, save the element value and its position to **cell_list**, zero out that element and adjacent overlapping elements, and then to repeat this procedure until all of the elements in **block** have been zeroed. Continuing with the listing: this is accomplished with the while loop, which will run until all of the elements in **block** are set to zero. **k** is used as an index, and is incremented for each loop iteration.

For each loop iteration, the maximum element in **block** is found by first treating **block** as a vector. **I** is the vector position of the element while **Y** is the element's value. The matrix position, **i** and **j** are then found from **I** and **blt**. The element's value and location are then added to **cell_list**. The elements representing overlapping blocks are then calculated, and the range is indicated by **i_min**, **i_max**, **j_min**, and **j_max**. (When finding these values, the edges of **block** must be considered.) The maximum element and overlapping elements are then set to zero.

Finally, **cell_list** is sized to eliminate the unused rows.

```
function cell_list = tile(M)
%     cell_list = tile(M)     This function takes as input a layer
%     from the QMF, and evaluates it using the 9 tile scheme.  The
%     output iscell_list, a three column list with the block energy,
%     the position in time, and the position in frequency.
%

% block dimensions
bf = 3;   % frequency dimension
bt = 3;   % time dimension

[m,n] = size(M);
blt = m-bt+1;
blf = n-bf+1;
block = zeros(blt,blf);

% compute the block energy
for j = 1:blf
  for i = 1:blt
    X = M([i:(i+bt-1)],[j:(j+bf-1)]);
    block(i,j) = sum(X(:));
  end;
end;
clear M;

% over-allocate space for cell_list
cell_list = zeros(length(block(:))./(bt.*bf),3);
```

```
% find the energy concentrations and list results
k = 0;
while max(block(:)) > 0

  k = k+1;

  % find the largest energy block
  [Y,I] = max(block(:));
  i = rem(I,blt);
  if i == 0
    i = blt;
  end;
  j = ceil(I./blt);

  % add entry to list
  cell_list(k,:) = [Y  i  j];

  % zero out overlapping blocks
  i_min = max(1,i-bt+1);  i_max = min(blt,i+(2 .*bt)-1);
  j_min = max(1,j-bf+1);  j_max = min(blf,j+(2 .*bf)-1);
  x = i_max - i_min + 1;
  y = j_max - j_min + 1;
  block([i_min:i_max],[j_min:j_max]) = zeros(x,y);

end; % while

cell_list = cell_list([1:k],:);
```

TILEMAX.M is similar to TILE.M, except it only finds the largest energy block (and takes far less time to run). This is particularly useful for some types of signal detection, where the signal/no signal decision is based on the largest amount of energy found in a block. The code is, literally, a cut and paste modification of the code in TILE.M.

```
function cell_max = tilemax(M)
%     cell_max = tilemax(M)    This is a short version of tile(M)
%     that computes only the largest energy cell in cell_list.
%
%     The tile(M) function takes as input a layer from
%     the QMF, and evaluates it using the 9 tile scheme.  The output
%     is cell_list, a three column list with the block energy, the
%     position in time, and the position in frequency.
%

% block dimensions
bf = 3;   % frequency dimension
bt = 3;   % time dimension

[m,n] = size(M);
blt = m-bt+1;
blf = n-bf+1;
block = zeros(blt,blf);
```

```
% compute the block energy
for j = 1:blf
  for i = 1:blt
    X = M([i:(i+bt-1)],[j:(j+bf-1)]);
    block(i,j) = sum(X(:));
  end;
end;
clear M;

% find the largest energy block
[Y,I] = max(block(:));
i = rem(I,blt);
if i == 0
  i = blt;
end;
j = ceil(I./blt);

% add entry to list
cell_max = [Y  i  j];
```

NTNOISE.M was used to generate Figure 5.13 in Chapter V.

```
%  NTNOISE.M  script file to generate empirically the noise
%             pdf curve (as a histogram)

N = 100;
x = [0:0.5:50]';  % histogram bins
y = zeros(size(x));

for loop = 1:N
      disp(loop)

      % generate noise waveform and find QMF bank output
      noise = noisegen(loop);
      qmf(noise,'con22',6);

      load layer6.dat;
      dat = layer6 .^2;
      clear layer6;

      % find block cell energy
      cell_list = tile(dat);

      % genrate histogram
      y = hist(cell_list(:,1),x)' + y;
end;

% display histogram
NB = sum(y);
y = y./NB;
bar(x,y)
axis([0 50 0 max(y)])
```

QMF9DET.M decomposes a waveform with the QMF bank tree and evaluates the output both using the nine tile scheme, and as a radiometer/threshold receiver. In QMF9DET.M, each signal is analyzed with QMF.M, the results from layer six are collected for the observation bandwidth, squared, and stored in **dat**. The highest energy block is then found using TILEMAX.M and the result is compared to the vector of thresholds, **Th**. For the radiometer part of the simulation, the elements of **dat** are summed, and the result is compared to the vector of thresholds, **ThR**.

```
function [det,rad_det] = qmf9det(waveform,Th,ThR)
%       [det,rad_det] = qmf9det(waveform,Th,ThR)
%                       Uses the QMF and the 9 tile
%                       scheme to detect a signal
%
%       Th is the threshold, and may be a vector.
%       det is a binary output vector and is 1 if a signal is assumed
%       to be present.

qmf(waveform,'con22',6);

load layer6.dat
[m,n] = size(layer6);
dat = layer6 .^2;
clear layer6;
dat = dat(1:m,(n./4 + 1):(3.*n./4));  % only look at frequencies
                                      % w channels

% find highest energy block
cell_max = tilemax(dat);

% compare to threshold
det = cell_max(1,1) > Th(:)';

% find radiometer detection (for comparison)
sd = sum(dat(:));
rad_det = sd > ThR(:)';
```

SIM9DET.M simulates detection with the nine tile scheme and with a radiometer. Following the listing: 100 signals for each set are generated that are similar to the signals used in the FBC simulations. In this case however, we make no assumptions about alignment between the cells of the signal and the tiles of our analyzing filter bank. For this reason, a random **phase** offset, affecting the timing of the hops, are included when generating each signal. **phase** is a 100 element vector of random integers from 0 to 127. Similarly, the channelization for each signal was shifted randomly by shifting **lf** over the bandwidth of one channel (0.0078125 Hz). In order to assure the signal stayed within the observation bandwidth, only 31 channels are used, vice 32 for the FBC simulations. **lf** is a 100 element vector of random integers from 0 to 127. QMF9DET.M is then used to make the detection decision.

```
% sim9det   script file to simulate detection using the 9 tile scheme

start = clock;

ThR = [16800:5:17500];
Th = [30:0.2:50];
rep = 100;

% set random phase
seed4 = 200000;
rand('seed',seed4);
phase = floor(128*(rand(1,rep)));

% set random channelization
seed5 = 300000;
rand('seed',seed5);
lf = 0.125 + (.0078125).*rand(1,rep);

for loop = 1:rep
      disp(loop)
      % generate signal
      [signal,channel] = ...
  ffhth(800000+loop,500000+loop,67,0,lf(loop),31,25,128,phase(loop));
      noise = ncisegen(loop+100000);
      waveform = noise + (.559).*signal;

      [D(loop,:) DR(loop,:)] = qmf9det(waveform,Th,ThR);
end;

P = sum(D)./rep;
PR = sum(DR)./rep;

disp(etime(clock,start));
```

QMF9ML.M decomposes a waveform using the QMF bank tree, and evaluates the output from various layers using the nine tile scheme. The largest energy block obtained from all of the layers is then compared against the vector **Th**, and a 1-0 detection vector, **det**, is returned.

```
function det = qmf9ml(waveform,Th,l_layer,h_layer)
%      det = qmf9ml(waveform,Th,l_layer,h_layer)
%            Uses the QMF and the 9 tile
%            scheme to detect a signal over several layers
%
%      Th is the threshold, and may be a vector.
%      l_layer is the first (lowest) layer examined
%      h_layer is the last (highest--or farthest down the tree)
%              layer examined
%      det is a binary output vector and is 1 if a signal is assumed
%          to be present.
```

```
qmf(waveform,'con22',h_layer);

cm = 0;
for loop = l_layer:h_layer
  loopstr = int2str(loop);

  % load the layer data
  eval(['load layer',loopstr,'.dat']);
  [m,n] = eval(['size(layer',loopstr,')']);
  dat = eval(['layer',loopstr,' .^2']);
  eval(['clear layer',loopstr]);
  % only look at frequencies w channels
  dat = dat(1:m,(n./4 + 1):(3.*n./4));

  % find highest energy block
  cell_max = tilemax(dat);
  cm = max(cm,cell_max(1,1));
end;

% compare to threshold
det = cm > Th(:)';
```

SIM9ML.M is similar to SIM9DET.M, except that the evaluation is performed for multiple layers of the QMF bank tree output, through QMF9ML.M.

```
% sim9ml   script file to simulate detection over multiple layers of
%          the qmf bank, using the 9 tile scheme

Th = [20:.2:60];
rep = 50;

l_layer = 3;
h_layer = 10;

% set random phase
seed4 = 200000;
rand('seed',seed4);
phase = floor(128*(rand(1,rep)));

% set random channelization
seed5 = 300000;
rand('seed',seed5);
lf = 0.125 + (.0078125).*rand(1,rep);

for loop = 1:rep
      disp(loop)
      % generate signal
      [signal,channel] = ...
  ffhth(800000+loop,500000+loop,67,0,lf(loop),31,25,128,phase(loop));
      noise = noisegen(loop+100000);
      waveform = noise + (.791).*signal;

      D(loop,:) = qmf9ml(waveform,Th,l_layer,h_layer);
end;

P = sum(D)./rep;
```

The Matlab files to accomplish the feature extraction algorithm are BLOKLIST.M, FINDDIM.M, and STBLOKLS.M.

BLOKLIST.M finds the lists of blocks, using the nine tile scheme, for layers between the lower layer, **l_layer**, and upper layer, **h_layer**. It assumes the signal has already been decomposed with QMF.M, and loads the results which have been previously stored to the hard drive. As with the detection simulations, here we will assume the signal ranges between 0.125 and 0.375 Hz, and so the code only examines this range (with **dat**). The nine tile scheme list, for each layer, is found using TILE.M (described above), and all of the results are saved to **cl**. The first three columns of **cl** are from the output of TILE.M and are the block's energy, left most (earliest) position in time, and lower position in frequency, respectively. The fourth column is added by BLOKLIST.M and is the layer the block came from.

FINDDIM.M takes **cl** from BLOKLIST.M and computes the beginning and end times and lower and upper frequency limits for each block. The code is straight forward. Note the dimensions of the blocks (in tiles) are set by **bf** and **bt**, and must agree with the values set in TILE.M. Also, when computing the frequency parameters an offset of 0.125 is added to account for the fact we are only looking at signals in the band 0.125 to 0.375 Hz. The output of FINDDIM.M is a list, **block_list**, where the first column shows the energy in each block, the second column is the layer number, the third and fourth columns are, respectively, the lower and upper time limits of the block, and the fifth and sixth columns are, respectively, the lower and upper frequency limits.

STBLOKLS.M takes **block_list** from FINDDIM.M, finds the largest energy blocks, and eliminates overlapping lower energy blocks. Following the listing: It does this by first sorting the rows of **block_list** by the blocks' energy, in descending order, and saving the result in **sort_list**. The code then goes into a while loop which executes until **sort_list** is empty. For each iteration of this loop, a logic vector L of zeros is created, whose length is the number of rows in **sort_list**. **L** will be used to indicate which rows will be saved, and so **sort_list** will become smaller with each iteration. Continuing with the listing: The first row of **sort_list**, for each iteration, is saved to **tw1_list**, and the dimensions of this block are compared to all of the other blocks in **sort_list**. For blocks not overlapping, the corresponding element of **L** is set to one. Finally, the rows of **sort_list** corresponding to zero elements of **L** are eliminated. The output, **tw1_list**, is the list of blocks we desire. As with **block_list**, the first column shows the energy in each block, the second column is the layer number, the third and fourth columns are, respectively, the lower and upper time limits of the block, and the fifth and sixth columns are, respectively, the lower and upper frequency limits. In a receiver like the one shown in Figure 1.2, **tw1_list** would be the list output from the analyzer block to the classifier block.

```
function [cl] = bloklist(l_layer,h_layer)
%     [cl] = bloklist(l_layer,h_layer)
%
%     This uses TILE.M and finds cell lists
%     for layers from l_layer to h_layer.
%
%     The lists are concatenated, and output as
%     cl, a four column list.  The fourth column is
%     the layer number (the other three come from TILE.M)
%
%     Only the portions in each layer from 0.125 to
%     0.375 Hz are examined.
```

```
cl = [];
for loop = l_layer:h_layer
  loopstr = int2str(loop);
  disp(['collecting cell list for layer ',loopstr,])

  % load the layer data
  eval(['load layer',loopstr,'.dat']);
  [m,n] = eval(['size(layer',loopstr,')']);
  dat = eval(['layer',loopstr,' .^2']);
  eval(['clear layer',loopstr]);
  % only look at frequencies 0.125 to 0.375 Hz
  dat = dat(1:m,(n./4 + 1):(3.*n./4));

  % get the cell list
  clnew = tile(dat);

  % add the layer number and include clnew in cl
  clnew = [clnew loop.*ones(length(clnew),1)];
  cl = [cl;clnew];
end;


function [block_list] = finddim(cl)
%     [block_list] = finddim(cl)
%
%     This takes cl from BLOKLIST.M and computes the
%     beginning and end times for each block and their
%     lower and upper frequency limit
%
%     for each row, the order is:
%  energy   layer #   low time   hi time   low freq   hi freq

% block dimensions
bf = 3;
bt = 3;

N = length(cl);
n = [1:N]';

% preallocate & fill in known columns
block_list = zeros(N,6);
block_list(:,1) = cl(:,1);  % energy
block_list(:,2) = cl(:,4);  % layer

% find beginning time of block
block_list(n,3) = (cl(n,2)-1).*(2.^block_list(n,2));

% find end time of block
block_list(n,4) = block_list(n,3) + bt.*(2.^block_list(n,2)) - 1;

% find lower frequency
block_list(n,5) = (cl(n,3)-1)./(2.^block_list(n,2)).*(0.5) + 0.125;

% find upper frequency
block_list(n,6) = block_list(n,5) + bf./(2.^block_list(n,2)).*(0.5);
```

```
function [tw1_list] = stblokls(block_list)
%     [tw1_list] = stblokls(block_list)
%
%     This takes block_list from FINDDIM.M, and sorts
%     the rows of the list by energy in each block
%     (descending order).  It then looks for overlapping
%     blocks and eliminates the lower energy blocks of
%     any that do overlap.

tw1_list = [];

% sort block list based on energy
[Y,I] = sort(block_list(:,1));
I = flipud(I);
sort_list = block_list(I,:);

while sort_list ~= []
  % L is a logic vector
  L = zeros(length(sort_list(:,1)),1);

  % transfer the top row of sort_list to tw1_list
  tw1_list = [tw1_list;sort_list(1,:)];

  % eliminate overlapping blocks from sort_list
  L = ((sort_list(1,3) >= sort_list(:,4)) | ...  % time
      (sort_list(1,4) <= sort_list(:,3))) | ...
      ((sort_list(1,5) >= sort_list(:,6)) | ...  % freq
      (sort_list(1,6) <= sort_list(:,5)));
  sort_list = sort_list(L,:);
end;
```

TW1FE2.M simulates the feature extraction of hopped signals whose cells have time bandwidth products of unity. In a manner similar to that used in SIM9DET.M, 25 signals are generated for each set, each with a random phase and channelization. Inside the loop, the signal is generated, added to noise, and the cell center frequencies and times are saved in **cell**. The waveform is then decomposed using QMF.M and the (large) input vectors are cleared. The QMF bank tree outputs are then analyzed using BLOKLIST.M, FINDDIM.M, and STBLOCKS.M, as described above, resulting in a list, **tw1_list**, of blocks, their energies and positions in the time frequency plane. These entries include blocks containing signal cells, and blocks containing energy from cell sidelobes and noise. Of these, the largest energy blocks are taken and sorted by time--and these blocks were found to be the ones, generally, containing the cells. (This was done heuristically here. In the actual receiver, like the one shown in Figure 1.2, this would be the job of the classifier.) The positions of these blocks are then compared to the known cell positions and the differences are save in **delta_time**, and **delta_freq**. The energy of the blocks is also saved to **energy**.

```
% tw1fe2.m   script file to extract features from a TW = 1 signal

delta_time = [];
delta_freq = [];
energy = [];

l_layer = 5;
h_layer = 7;
```

```
rep = 25;

% set random phase
seed4 = 200000;
rand('seed',seed4);
phase = floor(128*(rand(1,rep)));

% set random channelization
seed5 = 300000;
rand('seed',seed5);
lf = 0.125 + (.0078125).*rand(1,rep);

for loop = 1:rep
  disp(loop)
  % generate signal
  [signal,channel,slots] = ...
  ffhth(800000+loop,500000+loop,67,0,lf(loop),31,25,128,phase(loop));
  noise = noisegen(100000+loop);
  waveform = noise + (2).*signal;

  % determine the first 10 cells' center freq's and times
  N = 10;
  n = [1:N]';
  cell = zeros(N,2);
  cell(n,2) = (0.0078125).*(channel(n)-1) + (0.0078125)./2 + ...
              lf(loop);
  cell(n,1) = (n-1).*(128).*(25) + (128).*(slots(n)-1) + 64 - ...
              phase(loop);

  qmf(waveform,'con22',h_layer);

  % remove the inputs that are no longer needed
  clear noise;
  clear signal;
  clear waveform;

  % find the lists of blocks for all of the layers
  [cl] = bloklist(l_layer,h_layer);

  % find the locations of the blocks
  [block_list] = finddim(cl);

  % sort the blocks and eliminate overlaps
  [twl_list] = stblokls(block_list);

  % determine statistics
  % take blocks of twl_list > 100
  L = twl_list(:,1) > 100;
  l = twl_list(L,:);
  % sort by time
  [Y,I] = sort(l(:,3));
  l = l(I,:);
  l = l(1:10,:);  % only look at first 10

  % find differences in time & freq
  delta_time = [delta_time; (l(:,3)+l(:,4))./2 - cell(:,1)];
  delta_freq = [delta_freq; (l(:,5)+l(:,6))./2 - cell(:,2)];

  energy = [energy; l(:,1)];
end;
```

## Appendix C: Matlab Files Used to Carry Out Simulations Presented in Chapter VI

### Table of Contents

RADSIM.M and QMFRAD.M are used to simulate the detection of DS signals with a radiometer/threshold detector, as shown in Figure 6.5. When the probability of false alarm is measured, the line generating **signal** is commented out.

```
% radsim   script file to simulate using the qmf bank as a radiometer

start = clock;

Th = [1000:5:1100];
rep = 100;

for loop = 1:rep
      disp(loop)
      % generate signal
      [signal] = ds(300000+loop,0,0.3359375,0.015625);
      noise = noisegen(loop+100000);
      waveform = noise + (.0887).*signal;

      det(loop,:) = qmfrad(waveform,Th);
end;

P = sum(det)./rep;

disp(etime(clock,start));


function det = qmfrad(waveform,Th,K)
%      det = qmfrad(waveform,Th,K)
%      Uses the QMF as a radiometer to evaluate a waveform
%
%      The signal is assumed to be DS, with a bandwidth of 0.015625
%      centered in the 22nd tile (in freq) of the 5th layer.
%
%      Th is the threshold on the radiometer, and may be a vector.
%      det is a binary output vector and is 1 if a signal is assumed
%      to be present.

qmf(waveform,'tsinc',5);

load layer5.dat
dat = sum(layer5(:,22).^2);
clear layer5;

% threshold
det = dat > Th;
```

The code to compute a rectangle list is DS_SIM.M. It calls: ISOLATE.M, LOCATE.M, and STRECTLS.M. Beginning with the listing of DS_SIM.M: The inputs are the **waveform**, which may contain WGN or a DS signal plus WGN, the observation time, **T**, and the one sided noise density, **No**. **waveform** is decomposed with the QMF bank tree to the eighth layer, and this layer is loaded and saved to **dat**. The eighth layer was picked as a compromise. A higher layer increases the resolution, but at the expense of computation time.

To minimize aliasing problems, the center frequencies of the DS signals in our simulations are restricted to [0.125 + B/2, 0.375 - B/2] Hz, and so **dat** is trimmed so we only consider the region from [0.125, 0.375] Hz. The energy in the tiles are saved to **en_dat** by squaring the elements of **dat**, and then the spectral vector, **sv**, is computed.

The range of rectangle sizes considered (in terms of the number of bins of the spectral vector) is set from **lr** to **hr**. The code then loops, with each iteration considering a single rectangle size. A convolution of the spectral vector and a rectangular window is performed, the result is trimmed to remove edge effects, and then saved as **x**. The elements of **x** represent the energy of the rectangle as it is slid across the spectral vector. ISOLATE.M is used to find the largest energy elements, and to eliminate the elements of overlapping rectangles. The results of this are returned as **y**, a vector of the surviving energy values, and **pos**, an equal sized vector indicating the original positions of the corresponding elements of **y**. Using this information, the rectangle size, and the number of elements in **x** (computed from **layer**), the file LOCATE.M computes the lower and upper normalized frequency edges for each rectangle.

The test statistic, **u**, is then computed for each element of **y**. **new_list** is then formed, where each row represents a rectangle and the columns represent, from left to right, the test statistic, the rectangle's energy, the rectangle size (in spectral vector bins), the lower frequency edge of the rectangle, and the upper frequency edge. **new_list** is then concatenated onto **list**.

Finally, after **list** is complete, it is fed to STRECTLS.M, which sorts it based on **u**, saving the results to **sort_list**. It then saves the rectangles with high values of **u** to its output **rect_list**, while throwing out overlapping rectangles.

```
function [rect_list] = ds_sim(waveform,T,No)
%     [rect_list] = ds_sim(waveform,T,No)
%
% Performs the detection and feature extraction of DS
% Signals as described in Chapter 6.  The output, rect_list,
% is a list of possible DS signals.  For each row, the order
% is:
% u     energy      # of tiles     low freq     high freq
%
% where u is the test statistic described in Chapter 6
% and # of tiles refers to the number of tiles used to
% form the rectangle.

% decompose waveform & retrieve last layer
layer = 8;
ls = int2str(layer);
qmf(waveform,'tsinc',layer);
eval(['load layer',ls,'.dat']);
eval(['dat = layer',ls,';']);
```

```
% form spectral vector from dat
[m,n] = size(dat);
dat = dat(1:m,(n./4+1):(3 .*n./4));
en_dat = dat.^2;
sv = sum(en_dat);

% size range of rectangles (in tiles)
lr = 6;
hr = 10;

list = [];
for rs = lr:hr
  rect = ones(1,rs);
  x = conv(sv,rect);
  x = x(rs:length(x)-rs+1);  % trim ends
  [y,pos] = isolate(x,rs-1); % find max energy rectangles
  % find the lower & higher frequency limits
  % for each rectangle
  [lf,hf] = locate(pos,rs,layer);

  % find the test statistic
  W = rs.*(0.5)./(2^layer);
  u = (y - No*T*W)./sqrt(W);

  % add items to list
  n = length(y);
  new_list = [u y rs.*ones(n,1) lf hf];
  list = [list; new_list];
end;

rect_list = strectls(list);



function [y,pos] = isolate(x,z)
%     [y,pos] = isolate(x,z)
%
%  This takes a row vector, x, finds the maximum sized
%  element, and zeros out z adjacent elements on either
%  side.  This is repeated until there are no non-zero
%  adjacent elements. The high valued elements and their
%  original positions in x are returned as the column
%  vectors y and pos.

i = 1;
while max(x) ~= 0
  [y(i,1),I] = max(x);

  ln = max(1,(I-z));
  hn = min(length(x),(I+z));
  x(ln:hn) = zeros(1,length(ln:hn));
  pos(i,1) = I;
  i = i+1;
end;
```

```
function [lf,hf] = locate(pos,rs,layer)
%     [lf,hf] = locate(pos,rs,layer)
%
%  Takes pos from ISOLATE.M and, with the rectangle
%  size, rs, and layer number, layer, finds the lower
%  and high frequency limits for each rectangle.

% assumes the spectral vector ranges from
% 0.125 to 0.375 Hz
offset = 0.125;

% find the resolution, in Hz, for the layer
res = (0.5)/(2^layer);

lf = (pos-1).*res + offset;

hf = lf + rs.*res;



function [rect_list] = strectls(list)
%     [rect_list] = strectls(list)
%
%     This takes list from DS_SIM.M, and sorts
%     the rows of the list by the test statistic for each
%     row (descending order).  It then looks for overlapping
%     rectangles and eliminates the lower energy rectangles
%     of any that do overlap.

rect_list = [];

% sort list based on test statistic
[Y,I] = sort(list(:,1));
I = flipud(I);
sort_list = list(I,:);

while sort_list ~= []
  % L is a logic vector
  L = zeros(length(sort_list),1);

  % transfer the top row of sort_list to rect_list
  rect_list = [rect_list;sort_list(1,:)];

  % eliminate overlapping rectangles from sort_list
  L = ((sort_list(1,4) >= sort_list(:,5)) | ...  % freq
      (sort_list(1,5) <= sort_list(:,4)));
  sort_list = sort_list(L,:);
end;
```

DSSIM.M simulates the detection of DS signals when the center frequency and bandwidth are unknown, using the algorithm described in Chapter VI. The code is similar to RADSIM.M. except DS_SIM.M is called to find **rect_list**. Also, the center frequency of the DS signal is randomly varied between 0.125 + B/2 and 0.375 - B/2 Hz.

```
% dssim   script file to simulate the detection of DS signals

No = 2;
T = 2^15;
B = 0.015625;
W = B;
Th = ([800:5:1500] - No*T*W)./sqrt(W);
rep = 100;

seed1 = 200000;
seed2 = 300000;
seed3 = 100000;

% generate center frequencies
rand('seed',seed1);
range = 0.25 - B;
fc = 0.125 + (B/2) + range.*rand(1,rep);

for loop = 1:rep
      disp(loop)
      % generate signal
      [signal] = ds(seed2+loop,0,fc(loop),B);
      noise = noisegen(loop+seed3);
      waveform = noise + (.0887).*signal;

      rect_list = ds_sim(waveform,2^15,2);

      % compare energy in first rectangle against thresholds
      det(loop,:) = rect_list(1,1) > Th;
end;

P = sum(det)./rep;
```

DSFESIM.M simulates the estimation of a DS signal's center frequency and bandwidth using the algorithm described in Chapter VI. This code is similar to DSSIM.M. In DSFESIM.M, however, the DS signal will always be present in the waveform to be examined. Because there is no detection decision, there is no threshold. Rather, as **rect_list** is found for each of the 100 runs per set, estimates of signal features are made for the highest energy rectangle. An estimate of the signal energy is made using (6.21) and saved in **Ec(loop)**. An estimate of the bandwidth in terms of number of adjacent bins are saved in **bw(loop)**. An estimate of the center frequency is made by averaging between the high and low frequency edges of the rectangle, and is saved in **fc_est(loop)**. From this, the difference between the estimate and true center frequency are found and saved in the vector **fc_err**.

```
% dsfesim   script file to simulate the detection of DS signals
start = clock;

No = 2;
T = 2^15;
B = 0.015625;
W = B;

rep = 100;

seed1 = 2000;
seed2 = 3000;
seed3 = 1000;

% generate center frequencies
rand('seed',seed1);
range = 0.25 - B;
fc = 0.125 + (B/2) + range.*rand(1,rep);

for loop = 1:rep
      disp(loop)
      % generate signal
      [signal] = ds(seed2+loop,0,fc(loop),B);
      noise = noisegen(loop+seed3);
      waveform = noise + (.3545).*signal;

      rect_list = ds_sim(waveform,2^15,2);

      Noise = No*T*(rect_list(1,3)*(0.001953125));
      Ec(loop) = rect_list(1,2) - Noise;
      bw(loop) = rect_list(1,3);
      fc_est(loop) = (rect_list(1,4) + rect_list(1,5))/2;
end;

fc_err = fc - fc_est;
disp(etime(clock,start));
```

## Appendix D: Matlab Files Used to Carry Out Simulations Presented in Chapter VII

### Table of Contents

SIMFDDET.M is a script file used to simulate detection of hopped/DS signals. As with simulations described in earlier chapters, 100 runs were made for each set, first with noise alone to determine the probability of false alarm, and then with a signal added to determine the probability of detection. In addition to determining data for the block algorithm, the code also finds experimental ROC values for a radiometer covering the entire region of interest in the time frequency plane, and for a radiometer matched to the cell's position. These results can then be used as a check on the mathematical results discussed in Chapter VII.

Following the listing for SIMFDDET.M: **rep** is the number of runs made, **A** is the signal amplitude, **B** the signal cells' bandwidth, and **T** the signal cells' duration. **TR**, **TB**, and **TS** are the thresholds for the radiometer matched to the cell, block algorithm, and radiometer covering the entire region of interest, respectively.

Continuing with the listing, a vector of cell center frequencies (one element for each run in the set), **fc**, is generated randomly in the range [0.125+B/2, 0.375-B/2]. This range is used to ensure the signal is not too close to the upper or lower frequency limits of the QMF bank tree, so the effects of aliasing are minimized. The program then loops, once for each run, and generates a single cell of a TH/DS signal using TH.M (described in Chapter IV). This is represented as **signal**, and the cell's position among the time hop slots is saved as **position**. **waveform** is then generated from **noise** and, when finding the probability of detection, **signal**. In the last line of the loop, the information is passed to FHDSDET.M which decides whether a signal is present.

```
% simfddet   script file to simulate detection of hopped/DS

start = clock;
rep = 100;
A = 0.3496;
B = 0.0305;
T = 818;
TR = [16000:10:17000];
TB = [260:325];
TS = [50:150];

% random seeds
seed1 = 1000;
seed2 = 2000;
seed3 = 3000;
seed4 = 4000;

% set center freq
fmin = 0.125 + B/2;
fmax = 0.375 - B/2;
rand('seed',seed4);
fc = (fmax-fmin).*rand(1,rep) + fmin;
```

```
for loop = 1:rep
  disp(loop)
  % generate signal
  [signal,position] = ...
    th(seed1+loop,seed2+loop,0,fc(loop),40,818,0,0.0305);
  noise = noisegen(loop+seed3);
  waveform = noise + A.*signal;

  [DR(loop,:),DB(loop,:),DS(loop,:)] = ...
    fhdsdet(waveform,TR,TB,TS,fc(loop),position,B,T);
end;

PR = sum(DR)./rep;
PB = sum(DB)./rep;
PS = sum(DS)./rep;

disp(etime(clock,start));
```

Detection decisions for 1) a radiometer covering the entire region of interest in the time frequency plane, 2) a radiometer just covering the cell's position, and 3) the block algorithm, are made with FHDSDET.M. Following the listing: The inputs are **waveform**, **TR**, **TB**, **TS**, **fc**, **position**, **B**, and **T**. Inside the function, **waveform** is decomposed using QMF.M, and the seventh layer is loaded. (As we described above, any layer would do, as long as the tiles are smaller than the signal cell in both time and frequency.) The elements from the seventh layer are squared, and only the range from [0.125, 0.375] Hz is saved to **dat**.

The highest energy block is then found using TILEMAX.M, described in Chapter V and listed in Appendix B. In the case of all of the simulations in this chapter, the size of the blocks were set so **bt** = 13 (1664 seconds) and **bf** = 16 (0.0625 Hz), approximately two times the cell size in each dimension. The result from TILEMAX.M is then compared to the threshold vector **TB**, and the results of this comparison are saved in the output vector, **DB**.

All of the elements in **dat** are then summed to obtain the total energy collected by a radiometer set to the observation period with a bandpass filter covering [0.125, 0.375] Hz, and the results are compared to **TR**. The results of this comparison are then saved to **DR**.

**fc**, **position**, **B**, and **T**, are then used to compute the edges of the signal cell (in seconds and hertz). Then, the tile indices of those edges for layer seven are computed as **yh**, **yl**, **xh**, and **xl**. Finally, the relevant elements from **dat** are saved to **cell**, summed, and compared to **TS**. The results are saved to **DS**.

```
function [DR,DB,DS] = fhdsdet(waveform,TR,TB,TS,fc,position,B,T)
%
%     [DR,DB,DS] = fhdsdet(waveform,TR,TB,TS,fc,position,B,T)
%
%  This takes the input waveform, decomposes it with QMF.M,
%  and uses the block algorithm to find the highest energy
%  block (with TILEMAX.M).  The energy in this block is then
%  compared to a vector of thresholds, TB, to create a vector
%  of detection decisions, DB.
%
```

```
%  The position of the cell (in seconds and hertz) are also
%  found from the inputs, fc (center freq), position (in time),
%  B (the cell's bandwidth), and T (the cell's duration).  This
%  is then used to find the energy in the layer 7 decomposition
%  corresponding to the cell position, and the result is compared
%  to a threshold of vectors, TR, to create a vector of detection
%  decisions, DR.
%
%  Finally, the energy from the time frequency plane in the
%  observation period within [0.125,0.375] Hz are found and
%  compared to the vector of thresholds, TS, to create a vector
%  of detection decisions, DS.

% decompose the signal
qmf(waveform,'tsinc',7);

% load data
load layer7.dat
[m,n] = size(layer7);
dat = layer7 .^2;
clear layer7;
dat = dat(1:m,(n./4 + 1):(3.*n./4));
[m,n] = size(dat);

% find highest energy block & compare to threshold
cell_max = tilemax(dat);
DB = cell_max(1,1) > TB(:)';

% find radiometer detection
sd = sum(dat(:));
DR = sd > TR(:)';

% find the edges of the cell (freq & time)
fh = fc + B/2;
fl = fc - B/2;
tl = T*(position(1)-1) + 1;
th = T*(position(1)-1) + T;

% find the tile dimensions
layer = 7;
yh = min(n,floor((fh - 0.125)*2^(layer+1) + 1));
yl = max(1,floor((fl - 0.125)*2^(layer+1) + 1));
xh = min(m,ceil((th+1)/(2^layer)));
xl = max(1,ceil((tl+1)/(2^layer)));

% find the energy in that portion of t/f plane
cell = dat([xl:xh],[yl:yh]);
cd = sum(cell(:));

DS = cd > TS(:)';
```

The script file SIMSFDET.M is similar to SIMFDDET.M, and is used to simulate the detection of Slow FH signals. Since we are only interested in using a single cell of the signal in each run, and since the only code for generating these signals (SFH.M) generates cells over the entire observation period, it is necessary for SIMSFDET.M to produce a "mask" vector consisting of zeros for the times when the signal should not be observed, and ones for the duration of the cell that should be observed. This mask is then multiplied by the Slow FH signal to obtain the desired effect. To do this, first the vector **position** is generated randomly, with each element representing the particular cell that will be seen for each run. Then, inside the loop, **mask** is created with elements of one positioned appropriately.

```
% simsfdet   script file to simulate detection of slow FH

rep = 100;
A = 0.3536;
B = 0.03125;
T = 800;
TR = [16000:10:17000];
TB = [260:325];
TS = [50:150];

% random seeds
seed1 = 1000;
seed2 = 2000;
seed3 = 3000;
seed4 = 4000;

% set center freq
fmin = 0.125 + B/2;
fmax = 0.375 - B/2;
rand('seed',seed4);
fc = (fmax-fmin).*rand(1,rep) + fmin;

% set cell position in time
rand('seed',seed1);
position = ceil(((2^15)/T-1).*rand(1,rep));

for loop = 1:rep
  disp(loop)
  % generate signal
  signal = sfh(1,seed2+loop,0,fc(loop)-B/2,1,T,0,5,5);
  % mask all but one cell of signal
  mask = zeros(2^15,1);
  mask([T*(position(loop)-1)+1:T*(position(loop)-1)+T]) = ...
    ones(T,1);
  signal = signal.*mask;

  noise = noisegen(loop+seed3);
  waveform = noise + A.*signal;

  [DR(loop,:),DB(loop,:),DS(loop,:)] = ...
    sfdet(waveform,TR,TB,TS,fc(loop),position(loop),B,T);
end;

PR = sum(DR)./rep;
PB = sum(DB)./rep;
PS = sum(DS)./rep;
```

SIMFDFE.M simulates the estimation of parameters of hopped/DS signal cells. Because of the amount of time the code takes to run, only 50 runs were made per set. In each run a waveform consisting of noise and a single hopped/DS signal cell is formed (in a manner similar to the detection simulation script file SIMFDDET.M) and passed to FHDSFE.M. Vectors of true center frequencies, fc, and center times, tc, of the signal cells are also computed for later comparison with the estimates. The signal cells in these simulations had the same dimensions as those for the detection of hopped/DS signals, but the energy for these was increased by ten times to $\varepsilon = 500$.

```
% simfdfe  script file to simulate feature extraction of FH/DS
signals

start = clock;
rep = 50;

% cell parameters
A = 0.3496*sqrt(10);
B = 0.0305;
T = 818;

% random seeds
seed1 = 1000;
seed2 = 2000;
seed3 = 3000;
seed4 = 4000;

% set center freq
fmin = 0.125 + B/2;
fmax = 0.375 - B/2;
rand('seed',seed4);
fc = (fmax-fmin).*rand(1,rep) + fmin;

for loop = 1:rep
  disp(loop)
  % generate signal
  [signal,position] = ...
    th(seed1+loop,seed2+loop,0,fc(loop),40,T,0,B);
  noise = noisegen(loop+seed3);
  waveform = noise + A.*signal;

  [energy(loop),lf(loop),hf(loop),lt(loop),ht(loop)] = ...
    fhdsfe(waveform);

  % find center of cell time
  tc(loop) = T*(position(1)-1) + T/2;
end;


disp(etime(clock,start));
```

SIMSFFE.M is similar to SIMFDFE.M, except it simulates the estimation of parameters of Slow FH cells. The waveforms are constructed in a manner similar to the detection simulation script file SIMSFDET.M, with the same cell dimensions. As above, the cell energy is increased so $\varepsilon = 500$.

```
% simsffe  script file to simulate feature extraction of slow FH
%          signal

start = clock;
rep = 50;

% cell parameters
A = 0.3536*sqrt(10);
B = 0.03125;
T = 800;

% random seeds
seed1 = 1000;
seed2 = 2000;
seed3 = 3000;
seed4 = 4000;

% set center freq
fmin = 0.125 + B/2;
fmax = 0.375 - B/2;
rand('seed',seed4);
fc = (fmax-fmin).*rand(1,rep) + fmin;

% set cell position in time
rand('seed',seed1);
position = ceil(((2^15)/T-1).*rand(1,rep));

for loop = 1:rep
  disp(loop)
  % generate signal
  signal = sfh(1,seed2+loop,0,fc(loop)-B/2,1,T,0,5,5);
  mask = zeros(2^15,1);
  mask([T*(position(loop)-1)+1:T*(position(loop)-1)+T]) = ...
    ones(T,1);
  signal = signal.*mask;

  noise = noisegen(loop+seed3);
  waveform = noise + A.*signal;

  [energy(loop),lf(loop),hf(loop),lt(loop),ht(loop)] = ...
    fhdsfe(waveform);
end;

% find center of cell time
tc(:) = T*(position(:)-1) + T/2;

disp(etime(clock,start));
```

Both script files above call FHDSFE.M. Following the listing: The noise energy is first set, and then the minimum cell dimensions that the algorithm will look for are set. The minimum cell duration, **lct** = 400 seconds, and minimum bandwidth, **lcf** = 0.015 Hz, were used for all simulations. The block size is then set with **bt** and **bf** to match the parameters set in TILEMAX.M. **bt** = 13 and **bf** = 16 were always used here. The QMF layers that will be used are then set. As in the detection simulations, the block algorithm was performed on layer seven (**m_layer**). In all of the simulation results reported on below, the lower layer, **l_layer**, used for the time estimates was four while the higher layer, **h_layer**, used for the frequency estimates was ten. (Other layers were also tried, with similar results.)

Continuing with the listing, the waveform is then decomposed, the **m_layer** loaded, energy computed and results trimmed to include only [0.125, 0.375] Hz (signal cells are restricted to this region). The block algorithm is then performed using TILEMAX.M, and the highest energy block, and information about its position, is returned. The block's position, first in terms of the **m_layer** indices, and then in terms of seconds and hertz, are then computed, and information is passed to TIMEDIM.M and SPECDIM.M whose job is to estimate the position and dimensions of the cell within the block. Finally, using these results, the cell's energy is computed. For some of the simulation runs described below, these last lines were commented out, and the block energy (from TILEMAX.M) was returned instead.

```
function [energy,lf,hf,lt,ht] = fhdsfe(waveform)
%     [energy,lf,hf,lt,ht] = fhdsfe(waveform)
%
%  This takes the input waveform, decomposes it with
%  QMF.M, performs a block detection on the result
%  (using m_layer as set in the code), saves the highest
%  energy block (using TILEMAX.M), and then finds the
%  block's dimensions (in seconds and hertz).
%
%  The function then uses TIMEDIM.M and SPECDIM.M to estimate
%  the cell's position in the block.
%
%  OUTPUTS:
%          lf      low frequency of cell estimate
%          hf      high frequency of cell estimate
%          lt      low time of cell estimate
%          ht      high time of cell estimate
%          energy  energy of tiles under cell estimate

No = 2;

% minimum cell size
lct = 400;
lcf = .015;

% block size (these should be set the same as in TILEMAX.M)
bt = 13;
bf = 16;

% QMF tree layers for time estimate, block algorithm,
% and frequency estimate, respectively
l_layer = 4;
m_layer = 7;
h_layer = 10;
```

```
% decompose signal
qmf(waveform,'tsinc',h_layer);

% load data
loopstr = int2str(m_layer);
eval(['load layer',loopstr,'.dat']);
[m,n] = eval(['size(layer',loopstr,')']);
dat = eval(['layer',loopstr,' .^2']);
eval(['clear layer',loopstr]);
dat = dat(1:m,(n./4 + 1):(3.*n./4));
[m,n] = size(dat);

% find highest energy block
cell_max = tilemax(dat);
energy = cell_max(1);
i = cell_max(2);
j = cell_max(3);

% compute block position (tile indicies)
xbl = i;
xbh = i+bt-1;
ybl = j;
ybh = j+bf-1;

% compute block position in sec and Hz
lbf = (ybl-1)/(2^(m_layer+1)) + 0.125;
hbf = ybh/(2^(m_layer+1)) + 0.125;
lbt = (xbl-1)*2^m_layer;
hbt = xbh*2^m_layer - 1;

% estimate the cell's position within the block
[lt,ht] = timedim(l_layer,lbf,hbf,lbt,hbt,lct,No);
[lf,hf] = specdim(h_layer,lbf,hbf,lbt,hbt,lcf,No);

% isolate cell and compute energy
yl = max(1,floor((lf-0.125)*2^(m_layer+1) + 1));
yh = min(n,floor((hf-0.125)*2^(m_layer+1) + 1));
xl = max(1,ceil((lt+1)/(2^m_layer)));
xh = min(m,ceil((ht+1)/(2^m_layer)));

% estimate the cell energy
cell = dat(xl:xh,yl:yh);
energy = sum(cell(:));
```

TIMEDIM.M and SPECDIM.M are responsible for determining the time and frequency parameters (respectively) of the cell embedded in the block. The code for these functions are similar, and so only the listing for TIMEDIM.M will be described here. The inputs are: The layer to be used in finding the temporal vector, **layer**; the block dimensions (in seconds and hertz), **lbf**, **hbf**, **lbt**, **hbt**; the lower limit on the cell size, **lct**; and the noise energy density, **No**. The data for **layer** is loaded, the energy is computed, and the results are trimmed to [0.125, 0.375] Hz. The resulting matrix, **dat**, is then further trimmed to include only the tiles of **layer** that include the block. From this, the temporal vector, **v**, is formed. The lower and upper range of cell sizes, **lr** and **hr**, in terms of the tiles' time dimensions, are then determined, with the lower range based on **lct**, while the upper range is set equal to the block's duration. Then, the block's bandwidth, **W**, in hertz, is found. This will be used when calculating test statistics.

**list** is then preallocated as a three column matrix. When filled, each row will consist of the test statistic value, and the beginning and end positions in time of the maximum result from each convolution. The code then loops. In each iteration, a rectangle size (one tile length larger than in the previous iteration) is computed, the convolution is performed, the results are trimmed, the maximum value of the result is found as **y**, and its position is found as **I**. The test statistic for these results, **u**, is then computed and placed in **list**.

When the loop is completed, **list** will be full. The maximum test statistic is then found and the second and third columns of that row of **list** are used to compute the outputs: The lower position of the cell, **l**, and the upper position, **h**, both in seconds.

```
function [l,h] = timedim(layer,lbf,hbf,lbt,hbt,lct,No)
%
%     [l,h] = timedim(layer,lbf,hbf,lbt,hbt,lct,No)
%
%  This function takes a block from the time frequency
%  plane, computes a "temporal vector", and then
%  convolves the vector with various sized rectangular
%  windows, computing a test statistic, and finding an
%  estimate for the beginning and end times for a signal
%  cell.
%
%  INPUTS
%     layer      the layer of the QMF bank tree
%                decomposition used
%     lbf,hbf    the block's lower and upper
%                frequencies (in hertz)
%     lbt,hbt    the block's beginning and ending
%                times (in seconds)
%     lct        the lowest sized cell that the function
%                looks for
%     No         noise variance (for test statistic)
%
%  OUTPUTS
%     l,h        estimate of cell's beginning and end time
```

```
% load data
loopstr = int2str(layer);
eval(['load layer',loopstr,'.dat']);
[m,n] = eval(['size(layer',loopstr,')']);
dat = eval(['layer',loopstr,' .^2']);
eval(['clear layer',loopstr]);
dat = dat(1:m,(n./4 + 1):(3.*n./4));
[m,n] = size(dat);

% Trim dat to block dimensions
yl = max(1,floor((lbf-0.125)*2^(layer+1) + 1));
yh = min(n,floor((hbf-0.125)*2^(layer+1) + 1));
xl = max(1,ceil((lbt+1)/(2^layer)));
xh = min(m,ceil((hbt+1)/(2^layer)));
dat = dat(xl:xh,yl:yh);

% Form temporal vector
if yh-yl > 0
  v = sum(dat')';
else
  v = dat';
end;

% find lower and upper range of cell sizes
% in number of tiles
lr = max(1,ceil(lct/(2^layer)));
hr = xh - xl + 1;

% find block bandwidth (hertz)
W = hbf-lbf;

% convolve temporal vector with various
% sized rectangles
list = zeros(hr-lr+1,3);
for rs = lr:hr
  rect = ones(1,rs);
  x = conv(v,rect);
  x = x(rs:length(x)-rs+1);
  % find maximum of convolution for particular sized
  % rectangle
  [y,I] = max(x);
  % compute the rectangles' size (in seconds)
  to = rs*(2^layer) - 1;
  % compute test statistic
  u = (y - No*to*W)/sqrt(to*W);
  % list the test stat, beginnning position in time,
  % and end position in time
  list(rs-lr+1,:) = [u xl+I-1 xl+I+rs-2];
end;

% find the list row with the largest test statistic
[u,I] = max(list(:,1));
list = list(I,:);

% compute cell positions in seconds
l = (list(2)-1)*2^layer;
h = list(3)*2^layer - 1;
```

```
function [l,h] = specdim(layer,lbf,hbf,lbt,hbt,lcf,No)
%
%     [l,h] = specdim(layer,lbf,hbf,lbt,hbt,lcf,No)
%
%  This function takes a block from the time frequency
%  plane, computes a "spectral vector", and then
%  convolves the vector with various sized rectangular
%  windows, computing a test statistic, and finding an
%  estimate for the lower and upper frequencies for a
%  signal cell.
%
%  INPUTS
%     layer     the layer of the QMF bank tree
%               decomposition used
%     lbf,hbf   the block's lower and upper
%               frequencies (in hertz)
%     lbt,hbt   the block's beginning and ending
%               times (in seconds)
%     lcf       the lowest sized cell that the function
%               looks for
%     No        noise variance (for test statistic)
%
%  OUTPUTS
%     l,h       estimate of cell's lower and upper frequencies

% load data
loopstr = int2str(layer);
eval(['load layer',loopstr,'.dat']);
[m,n] = eval(['size(layer',loopstr,')']);
dat = eval(['layer',loopstr,' .^2']);
eval(['clear layer',loopstr]);
dat = dat(1:m,(n./4 + 1):(3.*n./4));
[m,n] = size(dat);

% Trim dat to block dimensions
yl = max(1,floor((lbf-0.125)*2^(layer+1) + 1));
yh = min(n,floor((hbf-0.125)*2^(layer+1) + 1));
xl = max(1,ceil((lbt+1)/(2^layer)));
xh = min(m,ceil((hbt+1)/(2^layer)));
dat = dat(xl:xh,yl:yh);

% Form spectral vector
if xh-xl > 0
  v = sum(dat);
else
  v = dat';
end;

% find lower and upper range of cell sizes
% in number of tiles
lr = max(1,floor(lcf*(2^(layer+1))));
hr = yh - yl + 1;

% find block duration (seconds)
to = hbt-lbt+1;
```

```
% convolve spectral vector with various
% sized rectangles
list = zeros(hr-lr+1,3);
for rs = lr:hr
  rect = ones(1,rs);
  x = conv(v,rect);
  x = x(rs:length(x)-rs+1);
  % find maximum of convolution for particular
  % sized rectangle
  [y,I] = max(x);
  % compute rectangles' size (in hertz)
  W = rs/(2^(layer+1));
  % compute test statistic
  u = (y - No*to*W)/sqrt(to*W);
  % list the test stat, lower position in freq
  % and upper position in freq
  list(rs-lr+1,:) = [u y1+I-1 y1+I+rs-2];
end;

% find the list row with the largest test statistic
[u,I] = max(list(:,1));
list = list(I,:);

% compute the cell positions in hertz
l = (list(2)-1)/2^(layer+1) + 0.125;
h = list(3)/2^(layer+1) + 0.125;
```

## Appendix E: Matlab Files Used to Carry Out Simulations Presented in Chapter VIII

### Table of Contents

SIMFDCH.M is a script file used to generate waveforms containing noise and a single hopped/DS cell, and to test the waveform using the least square algorithm described in Chapter VIII. This file is virtually identical to SIMFDDET.M, described in Appendix D, except the LS2.M, described below, is called to perform the analysis.

```
% simfdch8  script file to distinguish hopped/DS signals

start = clock;
rep = 50;

% cell parameters
A = 0.3496*sqrt(10);
B = 0.0305;
T = 818;

% random seeds
seed1 = 1000;
seed2 = 2000;
seed3 = 3000;
seed4 = 4000;

% threshold
Th = [1:1000];

% set center freq
fmin = 0.125 + B/2;
fmax = 0.375 - B/2;
rand('seed',seed4);
fc = (fmax-fmin).*rand(1,rep) + fmin;

for loop = 1:rep
  disp(loop)
  % generate signal
  [signal,position] = ...
    th(seed1+loop,seed2+loop,0,fc(loop),40,T,0,B);
  noise = noisegen(loop+seed3);
  waveform = noise + A.*signal;

  [D(loop,:),Bfd_est(loop),fest(loop)] = ls2(waveform,Th);
end;

Pfa = sum(D)./rep;
posfd_est = fc - fest;

disp(etime(clock,start));
```

SIMSFCH.M is a script file used to generate waveforms containing noise and a single hopped/DS cell, and to test the waveform using the least square algorithm described in Chapter VIII. This file is virtually identical to SIMFDDET.M, described in Appendix D, except the LS2.M, described below, is called to perform the analysis.

```
% simsfch8  script file to distinguish slow FH cells

start = clock;
rep = 50;
A = 0.3536*sqrt(10);
B = 0.03125;
T = 800;

% random seeds
seed1 = 1000;
seed2 = 2000;
seed3 = 3000;
seed4 = 4000;

% threshold
Th = [1:1000];

% set center freq
fmin = 0.125 + B/2;
fmax = 0.375 - B/2;
rand('seed',seed4);
fc = (fmax-fmin).*rand(1,rep) + fmin;

% set cell position in time
rand('seed',seed1);
position = ceil(((2^15)/T-1).*rand(1,rep));

for loop = 1:rep
  disp(loop)
  % generate signal
  signal = sfh(1,seed2+loop,0,fc(loop)-B/2,1,T,0,5,5);
  mask = zeros(2^15,1);
  mask([T*(position(loop)-1)+1:T*(position(loop)-1)+T]) = ...
    ones(T,1);
  signal = signal.*mask;

  noise = noisegen(loop+seed3);
  waveform = noise + A.*signal;

  [D(loop,:),Bsf_est(loop),fest(loop)] = ls2(waveform,Th);
end;

Pd = sum(D)./rep;
possf_est = fc - fest;

disp(etime(clock,start));
```

LS2.M performs the least square algorithm described in Chapter VIII. The first portion of the code decomposes **waveform** and performs the block detection algorithm using the **m_layer** output from the QMF bank tree. This code is block copied from the relevant portions of FHDSFE.M, described in Appendix D.

Once the maximum energy block is found, the **h_layer** QMF bank tree output is loaded, the portion including the block is saved to **dat**, and the spectral vector, **v**, is formed. From this a square matrix, **Mv**, is formed, all of whose rows are equal to **v**. Then, the lower and upper range of bandwidths, **lr** and **hr**, are computed, the block duration, **to**, tile bandwidth, **wb**, and the total energy in the spectral vector, **Ehat**, are found.

The code then loops, with the estimated bandwidth changed for each iteration. Inside the loop, the expected energy in the spectral vector is computed as **v_est**. (**v_est** is actually twice as long as **v**, so that the two vectors may be shifted and compared, without having to pad **v_est** with zeros.) A toeplitz matrix, **Mest**, is then formed, with the first row equal to **v_est**, and subsequent rows shifted one position to the right. A square portion of this matrix is then taken, so the result will have the same dimensions as **Mv**, and so the largest element of **v_est** is the diagonal element. The denominator of (8.6) is then formed as **den**, and a toeplitz matrix, **Md**, is formed in a manner similar to that described above. The test statistic **z** is then found (for the particular bandwidth then being considered in the loop), and saved to **list**, as are the bandwidth and the estimated center frequency.

After the loop has completed, the minimum of the test statistics is found, and that is compared to the vector of thresholds **Th**. A determination of whether the cell is hopped/DS is made and the results are saved to the output vector **D**. The bandwidth and center frequency estimates that yielded the minimum test statistic are also output, as **Best** and **fest** respectively.

```
function [D,Best,fest] = ls2(waveform,Th)
%     [D,Best,fest] = ls2(waveform,Th)
%
%     This takes the input waveform, decomposes it with QMF.M,
%     performs the least square algorithm described in
%     Chapter VIII, and then compares the test statistic against
%     a vector of thresholds, Th.
%
%     OUTPUTS:
%           D     vector of decisions (0 = hopped/DS cell;
%                                      1 = other cell)
%           Best  estimate of cell bandwidth
%           fest  estimate of center frequency

No = 2;

% minimum cell bandwidth
lcf = .015;

% block size (these should be set the same as in TILEMAX.M)
bt = 13;
bf = 16;
```

```
% QMF tree layers for decomposition and frequency estimate
m_layer = 7;
h_layer = 10;

% decompose signal
qmf(waveform,'tsinc',h_layer);

% load data
loopstr = int2str(m_layer);
eval(['load layer',loopstr,'.dat']);
[m,n] = eval(['size(layer',loopstr,')']);
dat = eval(['layer',loopstr,' .^2']);
eval(['clear layer',loopstr]);
dat = dat(1:m,(n./4 + 1):(3.*n./4));
[m,n] = size(dat);

% find highest energy block
cell_max = tilemax(dat);
energy = cell_max(1);
i = cell_max(2);
j = cell_max(3);

% compute block position (tile indicies)
xbl = i;
xbh = i+bt-1;
ybl = j;
ybh = j+bf-1;

% compute block position in sec and Hz
lbf = (ybl-1)/(2^(m_layer-1)) + 0.125;
hbf = ybh/(2^(m_layer+1)) + 0.125;
lbt = (xbl-1)*2^m_layer;
hbt = xbh*2^m_layer - 1;

% Now switch to higher layer

% load data
loopstr = int2str(h_layer);
eval(['load layer',loopstr,'.dat']);
[m,n] = eval(['size(layer',loopstr,')']);
dat = eval(['layer',loopstr,' .^2']);
eval(['clear layer',loopstr]);
dat = dat(1:m,(n./4 + 1):(3.*n./4));
[m,n] = size(dat);

% Trim dat to block dimensions
yl = max(1,floor((lbf-0.125)*2^(h_layer+1) + 1));
yh = min(n,floor((hbf-0.125)*2^(h_layer+1) + 1));
xl = max(1,ceil((lbt+1)/(2^h_layer)));
xh = min(m,ceil((hbt+1)/(2^h_layer)));
dat = dat(xl:xh,yl:yh);
```

```
% Form spectral vector
if xh-xl > 0
  v = sum(dat);
else
  v = dat';
end;
K = length(v);
k = 1:K;
% form square matrix
Mv = v;
for i = 2:K
  Mv = [Mv;v];
end;

% find lower and upper range of cell bandwidths
% in number of tiles
lr = max(1,floor(lcf*(2^(h_layer+1))));
hr = yh - yl + 1;

% find block duration (seconds)
to = hbt-lbt+1;

% find tile bandwidth (Hz)
Wb = (0.5)./(2.^h_layer);

% find measured total cell energy
Ehat = sum(v) - No*to*Wb*K;

% convolve spectral vector with various
% sized rectangles
list = zeros(hr-lr+1,3);
for rs = lr:hr
  % estimate energy vector
  Bhat = rs*Wb;
  Ebhat = Ehat.*(Wb/Bhat).*sinc_sq(Wb.*([1:(2*K)]-K)./Bhat);
  v_est = Ebhat + No*to*Wb;
  % form matrix
  c = zeros(1,K);
  c(1) = v_est(1);
  Mest = toeplitz(c,v_est);
  Mest = Mest(:,[K:(2*K-1)]);
  % form test statistic denomenator matrix
  den = 2*No*Ebhat + No^2*to*Wb;
  c(1) = den(1);
  Md = toeplitz(c,den);
  Md = Md(:,[K:(2*K-1)]);
  % find test statistic (for this bw)
  z = sum((((Mv-Mest).^2)./Md)');
  [list(rs-lr+1,1),pos] = min(z);
  list(rs-lr+1,2) = Bhat;
  list(rs-lr+1,3) = (yl + pos -1).*Wb - Wb./2 + 0.125;
end;
[z,I] = min(list(:,1));
D = z > Th;
Best = list(I,2);
fest = list(I,3);
```

## Bibliography


[1] Akensu, A. N. "The Binomial QMF-Wavelet Transform for Multiresolution Signal Decomposition, "*IEEE Transactions on Signal Processing. Vol. 41, No. 1*: 13-19 (January 1993).

[2] Arnold, B. C. and N. Balakrishnan. *Relations, Bounds, and Approximations For Order Statistics.* Berlin: Springer-Verlag, 1989.

[3] Arnold, B. C., N. Balakrishnan, and H. N. Nagaraja. *A First Course in Order Statistics.* New York: John Wiley & Sons, 1992.

[4] Auslander, L., C. Buffalano, R. Orr, and R. Tolimieri. "A Comparison of the Gabor and Short-Time Fourier Transforms For Signal Detection and Feature Extraction in Noisy Environments," *SPIE Vol. 1348 Advanced Signal-Processing Algorithms, Architectures, and Implementations.* (1990).

[5] Balakrishnan, N. and A. C. Cohen. *Order Statistics and Inference, Estimation Methods.* San Diego, CA: Academic Press, 1991.

[6] Boashash, B. and P. O'Shea. "Time-Frequency Analysis Applied to Signaturing of Underwater Acoustic Signals," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP88*: 2817-2820 (1988).

[7] Chen, C. K. and J. H. Lee. "Design of Quadrature Mirror Filters with Linear Phase in the Frequency Domain," *IEEE Transactions on Circuits and Systems--II: Analog and Digital Signal Processing, Vol. 39, No. 9*: 593-605 (September 1992).

[8] Chen, C. K. and J. H. Lee. "Design of Linear-Phase Quadrature Mirror Filters with Powers-of-Two Coefficients," *IEEE Transactions on Circuits and Systems--II: Analog and Digital Signal Processing, Vol. 41, No. 7*: 445-455 (July 1994).

[9] Chen, T. and P. P. Vaidyanathan. "Multidimensional Multirate Filters and Filter Banks Derived from One-Dimensional Filters," *IEEE Transactions on Signal Processing, Vol. 41, No. 5*: 1749-1765 (May 1993).

[10] Chui, C. K. *An Introduction to Wavelets.* San Diego, CA: Academic Press, 1992.

[11] Cohen, L. "Time Frequency Distributions - A Review," *Proceedings of the IEEE, Vol. 77, No. 7*: 941-981 (July 1989).

[12] Daubechies, I. "Orthonormal Bases of Compactly Supported Wavelets," *Communications on Pure and Applied Mathematics, Vol. XLI*: 909-996 (1988).

[13] David, H. A. *Order Statistics, Second Edition.* New York: John Wiley & Sons, 1981.

[14] Devore, J. L. *Probability and Statistics For Engineering and the Sciences.* Monterey, CA: Brooks/Cole Publishing Company, 1982.

[15] Dillard, R. A. "Detectability of Spread-Spectrum Signals," *IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-15, No. 4*: 526-537 (July 1979).

[16] Dillard, R. A. and G. M. Dillard. *Detectability of Spread Spectrum Signals*. Norwood, MA: Artech House, 1989.

[17] Dixon, R. C. *Spread Spectrum Systems*. New York: John Wiley & Sons, 1976.

[18] Engler, H. F. and D. H. Howard. *A Compendium of Analytic Models for Coherent and Noncoherent Receivers*. Avionics Laboratory, Air Force Wright Aeronautical Laboratories. Report Number: AFWAL-TR-85-1118. Wright-Patterson Air Force Base, OH, September 1985.

[19] Farrell, T. C., G. Prescott and S. Chakrabarti. "A Potential Use For Artificial Neural Networks in the Detection of Frequency Hopped Low Probability of Intercept Signals," *IEEE Wichita Conference on Communications, Networking and Signal Processing*: 25-30 (April 1994).

[20] Flandrin, P. "Time-Frequency Receivers For Locally Optimum Detection," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP88*: 2725-2728 (1988).

[21] Foreign Broadcast Information Service. "Characteristics of VHF and UHF Television Systems," Appendix to *Broadcasting Stations of the World, Part IV, 25th Ed.* Washington: Foreign Broadcast Information Service (United States Government), July 1972.

[22] Gardner, W. A. "Measurement of Spectral Correlation," *IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-34, No. 5*: 1111-1123 (October 1986).

[23] Gardner, W. A. and C. M. Spooner. "Signal Interception: Performance Advantages of Cyclic-Feature Detectors," *Department of Electrical Engineering and Computer Science, University of California, Davis.* (April 1991).

[24] Herley, C., J. Kovacevic, K. Ramchandran, and M. Vetterli. "Tilings of the Time Frequency Plane: Construction of Arbitrary Orthogonal Bases and Fast Tiling Algorithms," *IEEE Transactions on Signal Processing, Vol. 41, No. 12*: 3341-3359 (December 1993).

[25] Kay, S. and G. F. Boudreaux-Bartels. "On the Optimality of the Wigner Distribution for Detection," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP85*: 27.2.1-27.2.4 (1985).

[26] Koilpillai, R. D. and P. P. Vaidyanathan. "Cosine-Modulated FIR Filter Banks Satisfying Perfect Reconstruction," *IEEE Transactions on Signal Processing, Vol. 40, No. 4*: 770-783 (April 1992).

[27] Koilpillai, R. D. and P. P. Vaidyanathan. "A Spectral Factorization Approach to Pseudo-QMF Design," *IEEE Transactions on Signal Processing, Vol. 41, No. 1*: 82-92 (January 1993).

[28] Lathi, B. P. *Signals, Systems, and Communication*. New York: John Wiley & Sons, 1965.

[29] Leon-Garcia, A. *Probability and Random Processes For Electrical Engineering*. New York: Addison-Wesley, 1989.

[30] Levitt, B. K., U. Cheng, A. Polydorus, and M. K. Simon. "Optimum Detection of Slow Frequency-Hopped Signals," *IEEE Transactions on Communications, Vol. 42, No. 2/3/4*: 1990-1999 (February/March/April 1994).

[31] Lim, Y. C., R. H. Yang, and S. N. Koh. "The Design of Weighted Minimax Quadrature Mirror Filters," *IEEE Transactions on Signal Processing, Vol. 41, No. 5*: 1780-1789 (May 1993).

[32] Ludeman, L. C. *Fundamentals of Digital Signal Processing*. New York: Harper & Row, 1986.

[33] Marple Jr, S. L. *Digital Spectral Analysis With Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[34] Math Works, The. *The Student Edition of Matlab For MS-DOS Personal Computers*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[35] Nemsick, L. W. and E. Geraniotis. "Adaptive Multichannel Detection of Frequency-Hopping Signals," *IEEE Transactions on Communications, Vol. 40, No. 9*: 1502-1511 (September 1992).

[36] Oppenheim, A. V. and R. W. Schafer. *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[37] Porat, B. and B. Friedlander. "Performance Analysis of a Class of Transient Detection Algorithms--A Unified Framework," *IEEE Transactions on Signal Processing, Vol. 40, No. 10*: 2536-2546 (October 1992).

[38] Prescott, G. Course notes distributed in EENG 673, Applications of Communications Technology, Spread Spectrum Communications, Department of Electrical and Computer Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1988.

[39] Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in C, The Art of Scientific Computing, Second Edition*. New York: Cambridge University Press, 1992.

[40] Shanmugan, K. S. and A. M. Breipohl. *Random Signals: Detection, Estimation and Data Analysis*. New York: John Wiley & Sons, 1988.

[41] Torrieri, D. J. *Principles of Secure Communications Systems*. Norwood, MA: Artech House, 1985.

[42] Urkowitz, H. "Energy Detection of Unknown Deterministic Signals," *Proceedings of the IEEE, Vol. 55, No. 4*: 523-531 (April 1967).

[43] Vaidyanathan, P.P. *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[44] Vetterli, M. and C. Herley. "Wavelets and Filter Banks: Theory and Design," *IEEE Transactions on Signal Processing, Vol. 40, No. 9*: 2207-2232 (September 1992).

[45] Woodring, D. and J. D. Edell. "Detectability Calculation Techniques," Washington, DC: U. S. Naval Research Laboratory, September 1977.

[46] Ziemer, R. E., W. H. Tranter, and D. R. Fannin. *Signals and Systems: Continuous and Discrete, Third Edition*. New York: Macmillan, 1993.

[47] Zou, H. and A. H. Tewfik. "Parametrization of Compactly Supported Orthonormal Wavelets," *IEEE Transactions on Signal Processing, Vol. 41, No. 3*: 1428-1431 (March 1993).